

IMPLANTATION D'UN TERMINAL
SUR UN ORDINATEUR DE LA DEUXIEME GENERATION

THESE

présentée à la Faculté des Sciences
de l'Université de Genève
pour obtenir le grade de docteur ès sciences
mention informatique

par
Roger PIGUET
de
Genève

1973

THESE N° 1621

Genève

Imprimerie de la Section de Physique

1976

UNIVERSITE DE GENEVE

FACULTE DES SCIENCES

CENTRE INTERFACULTAIRE
DE CALCUL ELECTRONIQUE

PROFESSEUR J. HARMS

IMPLANTATION D'UN TERMINAL
SUR UN ORDINATEUR DE LA DEUXIEME GENERATION

THESE

présentée à la Faculté des Sciences
de l'Université de Genève
pour obtenir le grade de docteur ès sciences
mention informatique

par
Roger PIGUET
de
Genève

THESE N° 1621

Genève

Imprimerie de la Section de Physique

1976

La faculté des sciences, sur le préavis du professeur J. HARMS, Centre interfacultaire de calcul électronique, directeur de thèse, du professeur B. LEVRAT et du docteur E.M. PALANDRI (C.E.R.N.), autorise l'impression de la présente thèse, sans exprimer d'opinion sur les propositions qui y sont énoncées.

Genève, le 4 juillet 1973

Le Doyen:
Jean MULLER

Thèse n° 1621



Introduction

L'adjonction de nouveaux périphériques sur un ordinateur présente nécessairement deux aspects : celui de l'électronique et celui de la programmation. Ce travail est essentiellement consacré au second aspect.

L'implantation du terminal, construit par le groupe hardware du Centre Interfacultaire de Calcul Electronique, et son développement jusqu'à ce que toutes ses ressources soient utilisables, ont duré environ deux ans et demi. Pendant cette durée, trois activités principales ont été menées parallèlement :

- la mise au point des équipements, la recherche de diagnostics précis en cas de fonctionnement anormal pour faciliter le travail du groupe hardware;
- l'écriture des programmes permettant l'utilisation des équipements du terminal sous le système d'exploitation de l'ordinateur CDC 3800;
- le développement d'un ensemble de tests complets afin d'assurer la fiabilité de ces programmes.

Ce travail de thèse est divisé en 10 chapitres.

Non seulement il expose les programmes écrits pour le terminal, mais encore il présente, dans ses trois premiers chapitres, l'ordinateur CDC 3800 et le système d'exploitation DISK SCOPE, pour la partie concernant les entrées-sorties. Il peut ainsi servir de document de travail, complétant les manuels fournis par le constructeur, pour ceux qui effectueront d'autres développements sur cet ordinateur. Ces chapitres ne font en fait que reprendre le contenu des manuels de référence, en présentant toutefois les idées de base de manière plus approfondie. Des points, certes importants, mais pas en rapport direct avec le reste du travail, ont été laissés de côté, afin de ne pas alourdir inutilement le texte.

Le chapitre 4 aborde globalement les problèmes posés par l'implantation du terminal.

Le chapitre 5 décrit l'ensemble de programmes de tests, appelé système de simulation.

Les trois chapitres suivants examinent successivement chacun des nouveaux équipements. Leurs possibilités physiques et les programmes les dirigeant sont décrits séparément.

Le chapitre 9 passe en revue quelques programmes auxiliaires et les applications faites jusqu'à ce jour.

Enfin, le chapitre 10 étudie les extensions possibles. La configuration locale a permis quelques simplifications dans la programmation. Même si ces extensions ne sont pas réalisées, il est intéressant de faire ressortir les points qui peuvent être maintenus et ceux qui doivent être généralisés.

La rédaction d'un travail dans le domaine de la programmation de système pose passablement de difficultés. En effet, un système d'exploitation ne peut être étudié de manière linéaire. Il se compose d'un certain nombre de modules ayant certains liens entre eux. De même qu'il est impossible de voir un polyèdre d'un seul coup d'oeil, il est nécessaire de considérer successivement les différentes faces d'un système d'exploitation et d'en dégager les liens.

Dans le texte qui suit, les différentes faces ont été abordées dans un ordre aussi logique que possible; pour faire ressortir les liens, des renvois ou des répétitions volontaires ont été utilisées.

Une autre difficulté provient du nombre important de termes techniques anglais. Dans la plupart des cas où la chose a été possible, le terme français a été utilisé. Pour le reste, les mots anglais ont été employés tels quels dans le texte, des explications à leur sujet figurent dans l'annexe 3, d'autres termes sont définis dans le texte lui-même, lors de leur introduction.

Remerciements

Mes remerciements vont en premier lieu au professeur B. Levrat, directeur du Centre Interfacultaire de Calcul Electronique de l'Université de Genève, qui est l'initiateur du projet de développement d'un langage interactif. Je lui suis reconnaissant de m'avoir donné la possibilité de mener à chef ce travail et de la grande liberté qu'il m'a accordée dans sa réalisation.

Je remercie le professeur J. Harms qui a accepté de diriger ce travail. A son contact, j'ai pu me familiariser avec les méthodes de programmation de système, son aide m'a été particulièrement précieuse dans la détection et la correction d'erreurs dans le système d'exploitation et dans la définition de certaines options.

M. E.M. Palandri a accepté de faire partie du jury de ce travail et je l'en remercie.

Ma très vive gratitude va à M. J.F. Renevey, directeur du Centre Cantonal d'Informatique, et à M. P. Meylan, son adjoint, desquels dépend l'ordinateur CDC 3800. Ils ont été favorables au développement du projet et nous ont facilité le travail en nous accordant de nombreuses heures de temps machine, pendant l'exploitation normale et pendant les week-ends.

Je remercie les responsables de la compagnie CONTROL DATA qui nous ont accordé leur confiance et nous ont autorisé l'implantation d'un terminal de notre fabrication sur une de leurs machines.

L'aide des ingénieurs de cette compagnie m'a été précieuse, je suis particulièrement reconnaissant à MM. J.F. Champagnac et T. Bourgeois de leur grande disponibilité, ils ont apporté certaines précisions relatives au fonctionnement de l'ordinateur qui me faisaient défaut.

Je remercie également M. I. Nadas, analyste de cette même compagnie, qui m'a initié à la connaissance du système d'exploitation DISK SCOPE.

C'est grâce à l'aide financière du Fonds National que la construction de l'électronique a été possible, qu'il en soit ici remercié.

Le développement de tous les équipements a été l'oeuvre du groupe hardware du CICE, mes remerciements vont à G. Vuilleumier, responsable de ce groupe qui a également conçu et réalisé le dispositif de visualisation. Je remercie M. Wenger qui a conçu et mis au point l'électronique liée à la machine à écrire et le simulateur de canal. J.F. L'Haire a développé le clavier alphanumérique et de fonctions, avec l'aide de A. Delavy, il s'est chargé de la construction et du câblage de tout l'ensemble, je les en remercie. La réalisation d'un tel projet n'a pas pu se faire sans une étroite collaboration avec ce groupe, qui a toujours prêté attention à mes doléances. Une aide particulièrement sympathique m'est venue des chefs de travaux et du groupe des assistants du CICE, dans lequel a toujours régné un esprit excellent.

Pour l'utilisation du langage LISP sur le terminal, l'aide de X. Nguyen-Dinh m'a permis d'arriver rapidement à un résultat. J.P. Baud, qui a développé le langage INFOL interactif, a joué pour moi le rôle du principal utilisateur, par ses suggestions, il m'a permis d'améliorer la qualité du résultat. C. Pellegrini m'a apporté son aide dans les tests software sur la machine et sur les équipements, sa collaboration et son entrain m'ont permis de surmonter bien des difficultés. Son interpréteur BASIC a constitué un test supplémentaire pour mes programmes.

Aux autres membres du CICE, ainsi qu'aux opérateurs du CCI, qui m'ont rendu des services fort appréciés, va également ma reconnaissance.

Je remercie enfin Madame L. Noël et Mademoiselle M. Vermeire du soin et de la patience dont elles ont fait preuve lors de la dactylographie de ce texte, ainsi que le personnel de l'imprimerie qui en a facilité l'édition.

Genève, le 2 juin 1975.

Table des matières

Introduction	1
Remerciements	3
Table des matières	5
Chapitre 1 : Ordinateurs et systèmes d'exploitation	9
1.1 : Aperçu historique	9
1.2 : L'ordinateur CDC 3800	11
Chapitre 2 : Description de l'ordinateur CDC 3800	13
2.1 : Unité centrale et mémoire	13
2.2 : Système d'entrée-sortie	15
2.3 : Système d'interruption	20
2.4 : Configuration du système CDC 3800	23
Chapitre 3 : Opérations d'entrée-sortie sous le système d'exploitation DISK SCOPE	27
3.1 : Initialisation et construction du système	27
3.2 : Niveaux et tables des programmes	29
3.3 : Requêtes et unités logiques	31
3.4 : Tables	35
3.5 : Traitement d'une requête d'entrée-sortie au niveau 63 et dans INC	43
3.6 : Entrées et sorties d'un driver, routines à disposition des drivers	48
3.7 : Traitement type d'une requête d'entrée-sortie dans le driver	57
3.8 : Autres fonctions des drivers	58
Chapitre 4 : Généralités sur les drivers des équipements du terminal	61
4.1 : Fractionnement des opérations d'entrée-sortie	61
4.2 : Modifications des tables	63

Chapitre 5	: Le système de simulation	67
5.1	: Mise au point des équipements et des drivers	67
5.2	: Structure générale du système de simulation	71
5.3	: Le sous-programme ROUTINES	75
5.4	: Sous-programmes de simulation des équipements	78
Chapitre 6	: La machine à écrire	81
6.1	: Caractéristiques de l'équipement	81
6.2	: Le driver DRTWD	83
Chapitre 7	: Le dispositif de visualisation	89
7.1	: Caractéristiques de l'équipement	89
7.2	: Le driver DRDYD	96
7.3	: Contrôle de la mise en page	101
Chapitre 8	: Le clavier alphanumérique et de fonctions	105
8.1	: Caractéristiques de l'équipement	105
8.2	: Le driver DRKBD	107
8.3	: L'overlay KBDYL	115
Chapitre 9	: Utilisation des équipements interactifs	119
9.1	: Evaluation des performances	119
9.2	: Utilisation par un programme background	125
9.3	: Utilisation par un programme central	126
9.4	: Routines auxiliaires	128
9.5	: Utilisation par différents langages	129
Chapitre 10	: Extensions	133
10.1	: Modifications des équipements	133
10.2	: Implications pour le système d'exploitation	134
10.3	: Modifications des drivers	137
10.4	: Conclusion	141

Annexe 1	:	Quelques instructions COMPASS	143
Annexe 2	:	Les codes des caractères	145
Annexe 3	:	Termes anglais	151
Annexe 4	:	Références	153

Chapitre 1 : Ordinateurs et systèmes d'exploitation

1.1 Aperçu historique

Avec l'ordinateur, l'homme a pris l'habitude de tout classer, de tout hiérarchiser. Il n'est pas étonnant que les ordinateurs fassent également l'objet d'une classification désignée par l'expression "les trois générations d'ordinateurs". Cette classification ne fait pas l'unanimité et peut varier d'un auteur à l'autre.

Quel que soit le degré de complexité, les mêmes composants fondamentaux se retrouvent dans tous les ordinateurs : une unité centrale capable d'effectuer un certain nombre d'opérations élémentaires, une mémoire centrale et des équipements périphériques dont certains servent de mémoires de masse. Ces équipements périphériques sont généralement reliés à la mémoire par des canaux. La console joue un rôle important, elle permet de contrôler l'état de la machine, d'afficher le contenu des registres, d'entrer directement par des clés des instructions en langage machine, elle dispose généralement d'un bouton spécial de chargement automatique (load ou autoload) provoquant la lecture d'instructions sur un support physique déterminé ainsi que leur exécution.

La technologie de l'unité centrale est le premier des critères intervenant dans la classification en générations.

La première génération (qui commence avec les premiers ordinateurs vers 1940) est caractérisée par la présence de tubes électroniques, la deuxième génération (dès 1958) est marquée par l'apparition des circuits à transistors et la troisième (dès 1964) par celle des circuits intégrés. L'unité centrale n'a pas été la seule à évoluer, les performances des mémoires centrales et de tous les périphériques ont bénéficié de progrès tout aussi importants, ces éléments ne sont cependant pas déterminants dans cette classification. De même, la conception générale de l'ordinateur a évolué en vue d'une utilisation optimale de l'unité centrale. Les canaux dirigeant, indépendam-

ment de celle-ci, le transfert de données entre les périphériques et la mémoire, ainsi que les systèmes d'interruption sont deux éléments importants de cette évolution. Le développement du software est également un critère de classification en générations, correspondant approximativement à celles du hardware.

Dans la première génération, les programmes écrits en langage machine, directement ou par l'intermédiaire d'un programme traducteur appelé assembleur, étaient chargés à partir de la console.

La deuxième génération est marquée par l'apparition de programmes résidant en mémoire et facilitant le déroulement d'autres programmes : systèmes de contrôle des entrées-sorties, systèmes de compilation, moniteurs d'exécution. Ces différents programmes ont été groupés afin de former des systèmes d'exploitation, de plus en plus complexes, dont le but est de tirer le meilleur parti possible des ressources du hardware : le système d'exploitation reconnaît les cartes de contrôle, prend en charge les travaux et assure leur exécution, effectue toutes les opérations liées au système d'interruption, optimise l'utilisation des périphériques, met à la disposition du programmeur un certain nombre de services, détecte les conditions anormales, donne aux opérateurs la possibilité d'intervenir, tient une comptabilité des durées d'utilisation, etc. A cette époque apparaissent également les premiers langages supérieurs et leurs compilateurs.

Avec la troisième génération, les systèmes d'exploitation cherchent à améliorer encore les performances : à la monoprogrammation succèdent la multiprogrammation et le temps partagé. Les périphériques se dispersent et les accès à distance se développent. Grâce aux terminaux légers (claviers avec ou sans écrans de visualisation), le programmeur a l'impression de dialoguer avec l'ordinateur.

Actuellement, une quatrième génération, poussant encore plus loin les caractéristiques de la troisième est en voie d'apparition.

Il est évident qu'une classification en trois générations n'est pas suffisante pour un monde aussi vaste et varié que celui des

ordinateurs :

- à l'intérieur d'une même génération, le rapport des performances peut atteindre plusieurs centaines;
- les frontières ne sont pas bien définies : un ordinateur peut rassembler des caractéristiques de deux générations différentes.

En outre, il existe toujours, pour une installation, la possibilité d'être développée, c'est le cas tout particulièrement dans un centre de recherches ou universitaire. Des développements peuvent toucher tant le hardware que le software.

Ce travail présente essentiellement l'aspect software de l'implantation sur un ordinateur CDC 3800 d'un terminal comprenant une machine à écrire, un écran de visualisation et un clavier alphanumérique et de fonctions.

1.2 L'ordinateur CDC 3800

L'ordinateur CDC 3800 est le plus perfectionné de la série CDC 3000 supérieure, comprenant les modèles 3400, 3600 et 3800. Par ses caractéristiques hardware et software, il est un des ordinateurs les plus puissants de la deuxième génération, il a été conçu pour des applications scientifiques importantes. La production de cet ordinateur a commencé en 1963, alors que son prédécesseur CDC 3600 existait depuis deux ans.

Plusieurs systèmes d'exploitation ont été écrits pour les ordinateurs CDC 3600 - 3800.

Le premier est TAPE SCOPE dont les caractéristiques sont celles d'un système d'exploitation en monoprogrammation. Le nombre de boucles d'attente dans les opérations d'entrée-sortie est assez élevé du fait de la présence d'un seul programme à côté du superviseur.

La présence du système d'exploitation et de la bibliothèque sur une bande magnétique est un inconvénient majeur, qui allonge considérablement les boucles d'attente.

Dans la configuration de l'Université de Genève, ces boucles étaient encore allongées par le fait que le lecteur et le

perforateur de cartes ainsi que l'imprimante étaient en ligne. En fait TAPE SCOPE est plutôt conçu pour travailler avec des bandes magnétiques d'entrée et de sortie.

Introduit en 1965, le système d'exploitation DRUM SCOPE présente un progrès significatif. Le système DISK SCOPE, plus récent, ne présente aucune différence fondamentale avec DRUM SCOPE, seules quelques routines sont adaptées pour tenir compte des particularités du disque plutôt que de celles du tambour.

Les principaux avantages de DISK SCOPE sont :

- le système et la bibliothèque sont stockés sur disque;
- une possibilité restreinte de multiprogrammation existe grâce à la présence de programmes background;
- des queues d'entrée et de sortie sont constituées sur disque.

DISK SCOPE a été écrit de manière à maintenir autant que possible la compatibilité avec TAPE SCOPE. Des possibilités plus vastes sont offertes à l'utilisateur et quelques différences apparaissent dans les cartes de contrôle. C'est au niveau de la programmation en langage assembleur COMPASS qu'apparaissent le plus de différences.

Par rapport à TAPE SCOPE, l'amélioration des performances sous DISK SCOPE, pour l'ensemble des travaux, est de l'ordre de 30 à 35%.

Chapitre 2 : Description de l'ordinateur CDC 3800

2.1 Unité centrale et mémoire

L'unité centrale travaille sur des mots de 48 bits, conventionnellement numérotés de droite à gauche de 0 à 47. Ces mots peuvent être considérés comme des instructions, des quantités en virgule fixe, des quantités en virgule flottante ou comme des suites de caractères.

Les instructions machine, au nombre d'une centaine environ, peuvent avoir 24 bits (instructions courtes) ou 48 bits (instructions longues). Les instructions d'entrée-sortie sont décrites en détail ci-dessous, quelques autres instructions le sont dans l'annexe 1.

Dans le code interne de représentation des caractères, chacun d'eux est représenté par 6 bits, il s'agit donc d'un code à 64 caractères, par conséquent un mot peut en contenir 8.

Les principaux registres sont :

Le registre A, registre arithmétique principal, comporte 48 bits. Il effectue toutes les opérations arithmétiques, de décalage, de test, etc., soit indépendamment, soit avec l'aide du registre Q.

Le registre Q, registre arithmétique auxiliaire, comporte également 48 bits. Il effectue certaines opérations arithmétiques, de décalage, de test, etc. Dans certaines instructions (double précision, travail avec des masques), il complète le registre A.

Le registre D est un registre auxiliaire de 48 bits. Ses possibilités sont beaucoup plus limitées que celles de A et de Q. Les registres B1 à B6 sont des registres d'indice de 15 bits, ils permettent toutes les instructions d'indiquage.

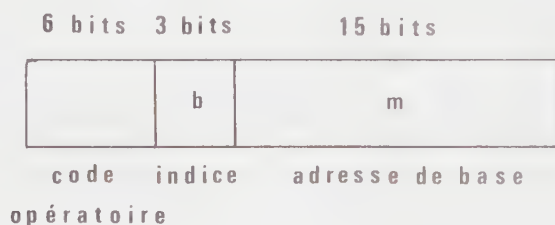
Le registre P est le compteur d'instructions, il est de 15 bits également.

Au moment de leur exécution, les instructions sont placées dans le registre U de 48 bits appelé registre d'instructions. La protection mémoire est assurée par un registre de 37 bits appelé bounds register (BR).

Mémoire et adressage

L'ordinateur CDC 3800 peut compter jusqu'à 8 blocs de mémoire appelés banks, contenant chacun 32768 mots de 48 bits.

Une instruction courte comportant une référence à la mémoire est de la forme :



La partie b, de 3 bits, a la signification suivante :

b = 0 : l'opération est effectuée sur l'adresse de base.

$1 \leq b \leq 6$: l'adresse de base est modifiée par le contenu du registre d'indice Bb.

b = 7 : signifie adressage indirect. L'évaluation de l'adresse se fait de la même manière sur les bits 0 à 17 de l'adresse m.

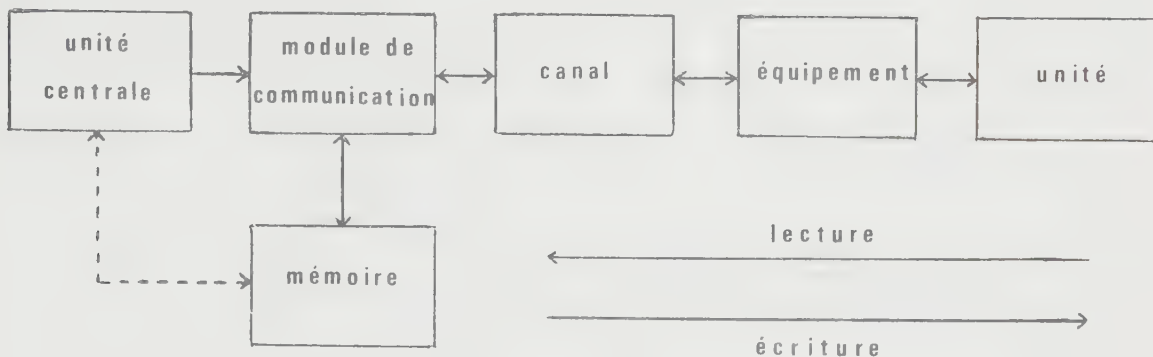
Ainsi, une instruction courte peut travailler sur n'importe lequel des 32768 = 2^{15} mots d'un bank. La configuration pouvant compter plusieurs banks, 2 registres supplémentaires, de 3 bits chacun, sont nécessaires.

L'operand bank register (OB) complète les 15 bits d'adresse donnés par l'instruction. Certaines instructions permettent de le modifier explicitement, tant qu'elles ne sont pas utilisées OB garde la même valeur.

L'instruction bank register (IB) complète le registre P, il spécifie le bank dans lequel se trouve l'instruction en cours d'exécution. Seules les instructions de branchement permettent de le modifier (voir annexe 1).

2.2 Système d'entrée-sortie

Les transferts entre la mémoire et les périphériques d'effectuent selon le schéma suivant.



Sur le signal de l'unité centrale, le module de communication déclenche une opération d'entrée-sortie. Ce module effectue les accès à la mémoire. Le canal (du type canal sélecteur) travaille de la même manière, quel que soit l'équipement auquel il est connecté. Il effectue, entre autres, l'assemblage de 4 bytes de 12 bits en un mot de 48 bits lors d'une lecture et le désassemblage d'un mot de 48 bits en 4 bytes de 12 bits lors d'une écriture. La vitesse de transfert est de 1.000.000 bytes/s au maximum. L'équipement (ou contrôleur) est la partie électronique qui fait le lien entre le canal et l'unité, il transmet exclusivement des bytes de 12 bits. L'unité est la partie qui travaille sur le support d'information, elle comporte généralement une partie mécanique.

Le module de communication et les canaux sont bidirectionnels, les équipements peuvent être mono ou bidirectionnels.

L'unité centrale peut avoir 4 modules de communication au maximum, chacun d'eux peut diriger 8 canaux et à chaque canal peuvent être rattachés 8 équipements. Le nombre d'unités par équipement varie de 1 à 16, selon le type d'équipement (1 lecteur de cartes, 8 disques, 16 bandes magnétiques, etc.). A un instant donné, le canal ne peut transférer des données que pour

un seul équipement; plusieurs canaux et plusieurs modules de communication, par contre, peuvent travailler simultanément.

Instructions d'entrée-sortie

Parmi le répertoire d'instructions figurent 7 instructions générales d'entrée-sortie, elles ne peuvent être exécutées que lorsque l'ordinateur est en mode superviseur (voir section 2.3).

CONN x,e,u,n (connect)

établit la connection canal x - équipement e - unité u, ou donne le contrôle à l'adresse de rejet n, en cas d'impossibilité.

EXTF x,c,n (external function)

exécute la fonction externe c sur l'unité connectée au canal x, ou sur le canal lui-même (un bit permet de faire la distinction), n est l'adresse de rejet.

Les fonctions sur le canal sont essentiellement des sélections et des libérations d'interruptions générées par le canal (voir section 2.3).

La signification des autres fonctions dépend des équipements, certaines fonctions ont un code dommun à tous les équipements. Elles peuvent être classées en 3 catégories :

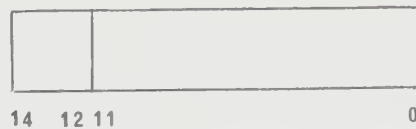
- déconnection (code 0);
- sélection et libération d'interruptions diverses sur l'équipement (voir section 2.3);
- exécution d'opérations généralement mécaniques et propres à une unité (saut de page, rebobinage d'une bande, etc.).

BEGR	}	x,(a)m,n	(begin read)
BEGW			(begin write)

Ces instructions initialisent un transfert (BEGR : lecture, BEGW : écriture) sur l'unité connectée au canal x, selon le mot de contrôle (voir ci-dessous) situé à l'adresse m dans le bank a. En cas de rejet, le contrôle est donné à l'adresse n.

COPY,CW,CWA x,b (copy status)

Cette instruction relève le statut du canal x et de l'équipement qui lui est connecté, dans le registre Bb. Les modificateurs CW (transfert du mot de contrôle en cours d'exécution dans A) et CWA (transfert de l'adresse du mot de contrôle suivant dans Q) sont optionnels. Le statut dans le registre Bb se présente comme suit :



Les bits 12 à 14 sont donnés par le canal, ils signifient :

bit 12 : une lecture est en cours;

bit 13 : une écriture est en cours;

bit 14 : erreur de parité ou de transmission détectée par le canal.

Les bits 0 à 11 sont donnés par l'équipement et l'unité, seuls les bits 0 et 1 ont une signification globale :

bit 0 : l'unité est prête (les conditions à remplir pour qu'une unité soit prête dépendent de son type, elles expriment le fait que l'unité est en mesure d'effectuer une opération d'entrée ou de sortie);

bit 1 : l'unité est occupée (là également, la définition précise est fonction du type d'unité, en gros, cela veut dire que l'unité est présentement en train d'effectuer une opération d'entrée-sortie ou une fonction).

Les autres bits sont liés à des interruptions, à des conditions normales (mode de fonctionnement, densité, etc.) ou anormales (fin de fichier, erreur de parité, bourrage, etc.) propres au type d'équipement.

CLCH x (clear channel)

efface toutes les interruptions et les sélections d'interruption sur le canal x, coupe toutes les opérations en cours sur ce même canal et le déconnecte.

Mots de contrôle

Les ensembles modules de communication - canaux peuvent être considérés comme de petits ordinateurs contrôlant les opérations d'entrée-sortie sans intervention de l'unité centrale après l'initialisation. Des mots de contrôle (terminologie IBM : programmes canal) sont utilisés pour diriger les transferts, ils peuvent être considérés comme des instructions à ces petits ordinateurs.

Lors de l'exécution d'une instruction BECR ou BECW, le mot de contrôle situé à l'adresse (a)m est transmis au module de communication. Ce mot de contrôle a l'allure suivante :



bits 0-17 : (a)m est l'adresse (avec terme de bank) du premier mot de la zone de données;

bits 24-38 : w est le nombre (en mots) de données.

Ces deux champs sont transmis à des registres dans le canal. Chaque fois que celui-ci transmet un mot de données, il augmente l'adresse d'une unité et diminue le compte de mots d'une unité. L'opération est terminée au plus tard lorsque ce compte est ramené à zéro.

Le bit 44, s'il vaut 1, provoque l'adressage indirect : le champ (a)m désigne non pas l'adresse de la première donnée, mais celle du mot de contrôle suivant. Le bit 44 est toujours le premier testé.

Le bit 45 signifie, s'il vaut 1, transmettre et sinon, ne pas transmettre.

Le bit 46, s'il vaut 1, fait intervenir la notion de fin d'enregistrement.

Le bit 47, s'il vaut 1, provoque l'enchaînement au mot de contrôle suivant (c'est-à-dire situé à l'adresse suivante) dès que le mot de contrôle en cours a été totalement exécuté.

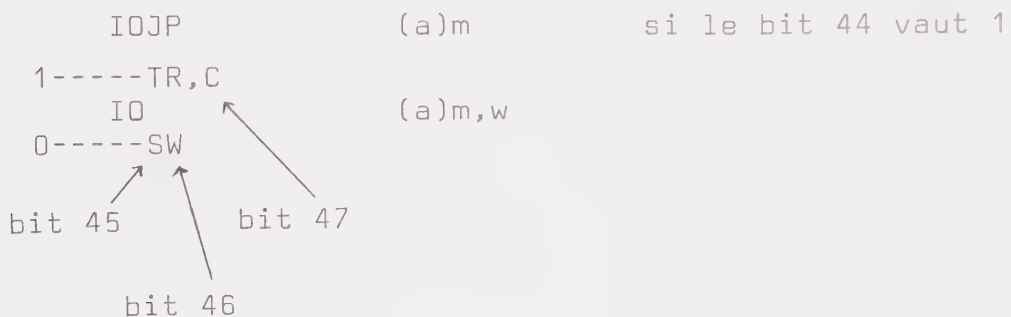
Le sens des bits 45 et 46 doit encore être précisé car il dépend du sens du transfert et de la présence du bit 47. En lecture la fin d'enregistrement a le même effet que le retour à zéro du compteur de mots si le bit 46 vaut 1.

47	46	45	
0	0	0	lire w mots sans les transmettre (sauter w mots)
0	0	1	lire w mots et les transmettre (transmettre w mots)
0	1	0	sauter un enregistrement de w mots au maximum
0	1	1	transmettre un enregistrement de w mots au maximum
1	0	0	sauter w mots
1	0	1	transmettre w mots
1	1	0	sauter un enregistrement de w mots au maximum et passer au mot de contrôle suivant
1	1	1	transmettre un enregistrement de w mots au maximum et passer au mot de contrôle suivant.

En écriture, si le bit 45 vaut 0, des mots ne contenant que des zéros sont transmis.

47	46	45	
0	0	0	écrire w mots de zéros et mettre une fin d'enregistrement
0	0	1	écrire w mots et mettre une fin d'enregistrement
0	1	0	écrire w mots de zéros et mettre une fin d'enregistrement
0	1	1	écrire w mots et mettre une fin d'enregistrement
1	0	0	écrire w mots de zéros et passer au mot de contrôle suivant
1	0	1	écrire w mots et passer au mot de contrôle suivant
1	1	0	écrire w mots de zéros, mettre une fin d'enregistrement et passer au mot de contrôle suivant
1	1	1	écrire w mots, mettre une fin d'enregistrement et passer au mot de contrôle suivant.

L'expression "mettre une fin d'enregistrement" est vague, elle dépend du périphérique envisagé. Le cas le plus simple est celui de la bande magnétique où un espace de 3/4 de pouce sépare deux enregistrements consécutifs. Sur les autres périphériques, l'interprétation de cette expression dépend du sens attribué à l'enregistrement physique (ligne, carte, etc.). Les mnémoniques attribués par l'assembleur aux mots de contrôle sont :



Exemple :

Chaîne de 3 mots de contrôle :

IOTW,C	(0)A,10
IOSW,C	(0)0,5
IOTR	(1)B,20

2.3 Système d'interruption

Deux registres de 48 bits travaillant en parallèle sont associés au système d'interruption : le registre d'interruption (IR) et le masque du registre d'interruption (IM).

Il existe 16 conditions internes d'interruption, à chacune d'elles est attaché un bit (entre 0 et 15) dans IR et IM. Les 32 autres bits correspondent à chacun des 32 canaux adaptables sur un système CDC 3800.

Une cause possible d'interruption provoque effectivement une interruption si les trois conditions suivantes sont remplies :

- le système d'interruption est actif;
- le bit associé à la condition vaut 1 dans IM;
- la cause se produit (ce qui met le bit correspondant de IR à 1).

Le produit logique de IR et IM est effectué automatiquement et testé entre chaque instruction et même à l'intérieur de certaines instructions comportant plusieurs pas.

Lors d'une interruption les opérations suivantes s'effectuent :

- le système d'interruption est mis en mode inactif, la machine est alors en mode d'interruption (terminologie IBM : mode superviseur);
- une instruction de transfert permettant de reprendre le contrôle du programme là où il a été interrompu est stockée à l'adresse 0 du bank 0;
- le contrôle est donné à l'adresse 1 du bank 0.

Le programme d'interruption doit sauver les registres, analyser les causes de l'interruption (chaque bit de IR peut être testé), les traiter et effacer à l'aide d'une instruction appropriée le bit mis à 1 dans IR.

Pendant le traitement d'une interruption, d'autres peuvent se produire, elles sont stockées dans IR mais n'ont pas d'autre effet puisque la machine est en mode d'interruption.

Le système d'interruption redevient actif

- par l'exécution de l'instruction machine appropriée,
- ou par l'exécution de l'instruction de transfert à l'adresse 0(0), pour autant qu'il n'y ait plus d'autre interruption (auquel cas, le contrôle serait redonné à l'adresse (0)1).

C'est ce deuxième procédé qui est usuellement employé.

Interruptions liées au système d'entrée-sortie.

Les bits 16 à 47 de IR et de IM sont liés aux canaux 0 à 31 respectivement. Les bits 16 à 47 de IM sont toujours à 1. Le bit, dans IR, lié à un canal peut être mis à 1 par plusieurs conditions indépendantes, certaines proviennent du canal, d'autres de ses équipements. A chacune de ces conditions sont attachés deux codes de fonctions utilisables par l'instruction EXTF.

Le premier sélectionne l'interruption sur la condition, c'est-à-dire demande l'envoi d'une interruption lorsque la condition se produira.

Le second libère l'interruption sur la condition, c'est-à-dire efface la condition d'interruption si elle est présente, ainsi que la sélection correspondante.

Un bit dans IR est remis à zéro

- par l'effacement individuel de chaque condition,
- ou par l'utilisation de l'instruction `IRCLR`.

Le premier procédé correspond au traitement usuel, le second est prévu pour des cas exceptionnels.

Conditions pouvant provoquer une interruption du canal :

- fin de la chaîne de mots de contrôle, c'est-à-dire fin de la transmission pour le canal;
- erreur (erreur de parité dans la transmission des données, dans un mot de contrôle, etc.).

Conditions pouvant provoquer une interruption de l'équipement (ces conditions varient d'un équipement à l'autre) :

- fin d'opération, c'est-à-dire fin de la transmission pour l'équipement;
- fin d'opération anormale (erreur de parité, fin de fichier, bourrage, etc.);
- unité disponible (ready and not busy) : cette interruption est liée à la partie mécanique de l'unité, elle signifie que l'unité est prête pour une nouvelle opération;
- toute autre condition liée à la nature de l'unité.

Une interruption peut avoir lieu que l'équipement soit connecté ou non (la ligne d'interruption est permanente). Pour une interruption marquant la fin d'un transfert, la connection est forcément présente. Par contre, ce n'est pas nécessairement le cas pour une interruption marquant la fin d'une opération déclenchée par une fonction seulement :

- saut de page, sur une imprimante;
- rebobinage d'une bande;
- caractère frappé sur une machine à écrire, etc.

L'instruction machine

CPJ

x

(channel product jump)

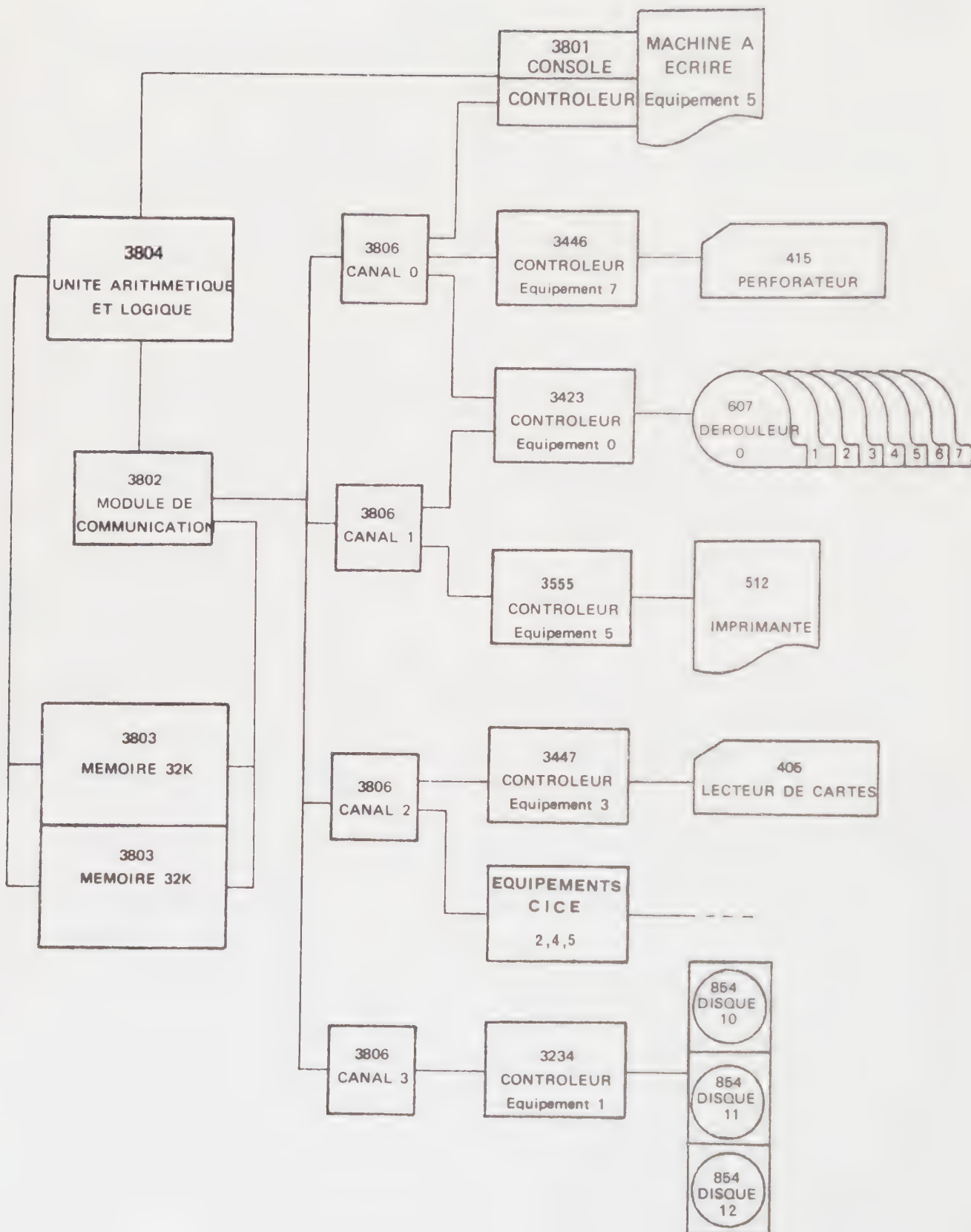
permet de déterminer la provenance d'une interruption sur un canal, elle donne toujours un et un seul des renseignements suivants :

- erreur de parité du mot de contrôle
 - erreur de parité dans la transmission
 - fin de chaîne
 - numéro de l'équipement si l'interruption a été générée par un équipement;
 - pas d'interruption sur le canal.
- } pour une interruption générée par le canal;

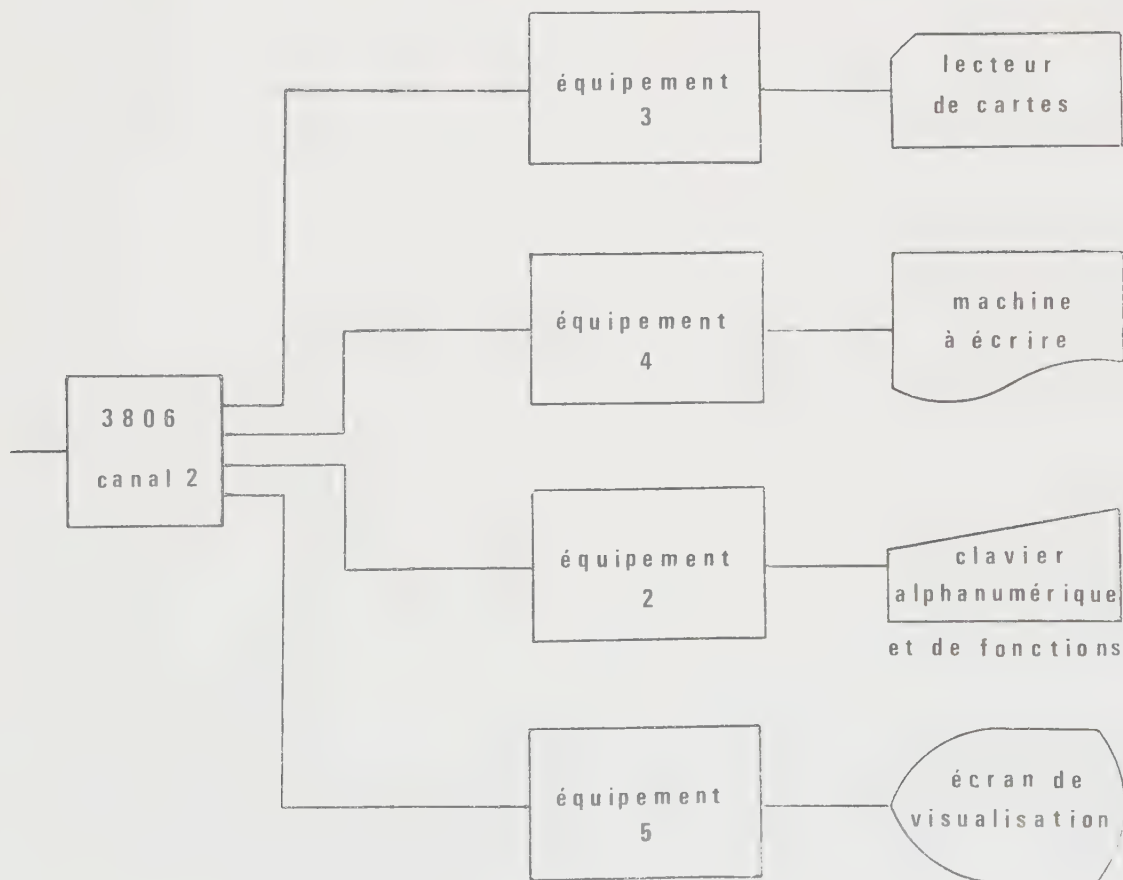
Par contre, au niveau de l'équipement, il n'y a pas moyen de connaître directement l'unité responsable d'une interruption. Il faut tester à l'aide de l'instruction COPY toutes les unités attachées à l'équipement, ou seulement celles sur lesquelles une interruption est en attente (ce qui nécessite l'emploi d'une table).

2.4 Configuration du système CDC 3800

Il est bien évident qu'un tel système est tout à fait modulaire et que ses constituants peuvent varier tant en nombre qu'en type. La configuration de la machine utilisée par l'Université de Genève est la suivante :



Les équipements CICE ont été conçus et construits au Centre interfacultaire de calcul électronique. Ils ne présentent pas tout à fait les caractéristiques des autres équipements fournis par le constructeur : ils possèdent en commun la partie électronique reliée au canal, qui comprend le contrôle de la parité et la sélection d'équipement. La restriction est donc qu'il serait impossible de débrancher l'un des équipements pour aller le brancher sur un autre canal. Elle est tout à fait admissible car un seul équipement peut utiliser le canal pour un transfert à un instant donné. A chaque équipement n'est attachée qu'une seule unité.



Chapitre 3 : Opérations d'entrée-sortie sous le système d'exploitation DISK SCOPE

3.1 Initialisation et construction du système

DISK SCOPE est initialisé à partir d'une bande magnétique, désignée désormais par le terme de "bande système".

Les principales étapes de cette opération sont :

- Grâce au processus d'autoload, la première routine écrite en code machine absolu sous forme d'un seul enregistrement est lue et exécutée. Elle comporte essentiellement deux petits programmes de contrôle (appelés drivers) pour la machine à écrire de la console et pour la bande magnétique, ainsi qu'un chargeur restreint.
- Ce chargeur provoque la lecture et le stockage de la deuxième routine appelée PREDRUM, écrite sous forme translatable (relocatable). PREDRUM est stocké dans la partie supérieure du bank 0 et reçoit le contrôle, lorsque sa lecture est terminée, pour construire en mémoire la partie résidente du système et préparer sur le disque les parties non-résidentes.
- Les premiers éléments que PREDRUM lit sont des tables décrivant la configuration et les paramètres du système, elles sont sur la bande sous forme d'images-cartes et sont décrites de manière symbolique. Ces tables sont traduites de la forme symbolique dans leur forme interne, celle qui est stockée en mémoire. Les trois premières tables sont PARAMS (décrivant les paramètres d'installation du système), HTL (donnant le nom et les types d'équipement) et AET (indiquant la configuration des périphériques). Ces deux dernières tables, ainsi que leur forme symbolique externe, sont décrites en détail dans la section 3.4.
- Viennent ensuite les routines du système proprement dit, commençant par la plus importante, RESIDENT. Elles sont toutes écrites sous forme translatable. PREDRUM les lit, effectue les opérations de chargement, construit les tables et les

répertoires et réécrit les routines sur disque.

A la fin de cette phase, une partie du système se trouve en mémoire et y restera en permanence : la routine RESIDENT, les tables et quelques autres sous-programmes.

- PREDRUM lit ensuite les programmes background (voir section 3.2) puis les routines de bibliothèque, les copie sur disque et construit les répertoires.
- L'initialisation finale du système est effectuée par la routine BOOT (après libération de la place occupée par PREDRUM) sous le contrôle de la partie résidente du système.
Pendant cette phase, un certain nombre de programmes d'utilisation courante (les compilateurs, par exemple) sont écrits sous forme absolue sur le disque pour permettre ultérieurement un chargement plus rapide.
- Après indication de la date et de l'heure par l'opérateur, le système est prêt à prendre les travaux en charge. En cas de situation anormale, il peut être rechargé à partir du disque.

Parmi les programmes faisant partie de la bibliothèque, il en est deux qui jouent un rôle particulièrement important, puisqu'ils sont utilisés pour construire ou modifier le système; ils ont abondamment servi à la réalisation de ce travail.

- L'assembleur COMPASS (comprehensive assembly system) offre en plus des mnémoniques relatifs à chaque instruction machine un jeu étendu de pseudo-instructions et des possibilités intéressantes de codage génératif (macros, répétition d'instructions, assemblage conditionnel, etc.). Il dispose d'autre part d'un éditeur travaillant au niveau de la carte et permettant la mise à jour aisée de programmes écrits sur bande. Le code généré par l'assembleur est appelé binaire translatable, il est généralement traité par LOADM, le chargeur principal du système.

L'annexe 1 donne les mnémoniques de quelques instructions en langage COMPASS, apparaissant dans ce travail.

- L'éditeur de bandes système LIBEDIT peut travailler tant sur des routines binaires (sous-programmes) que sur des routines BCD (tables, par exemple).

D'autre part, il dispose d'un petit chargeur permettant de transformer en code absolu une routine écrite en binaire translatable. Le rôle principal de ce chargeur est de fabriquer la première routine de la bande système, mais il a aussi servi à écrire en absolu des programmes contrôlant le fonctionnement des équipements et utilisant directement les instructions machine d'entrée-sortie, toutes inaccessibles à un programme sous le contrôle de DISK SCOPE (voir section 5.1).

3.2 Niveaux et tables des programmes

DISK SCOPE permet la présence simultanée de 64 programmes au maximum, numérotés de 0 à 63 selon l'ordre croissant de priorité. Ces niveaux ont des significations bien déterminées.

niveau	signification
0	programme central
1	routines d'interruption du programme central
2	exécutif
3	chargeur
4-61	programmes background
62	sous-système MSIO
63	traitement des requêtes

Les niveaux 0 à 3 constituent le groupe du programme central. L'utilisateur ne travaille qu'au niveau 0 et éventuellement au niveau 1, l'exécutif et le chargeur sont au service du programme central.

Les programmes background sont des petits programmes, servant essentiellement à faire des transferts d'une unité à une autre. Ce sont eux qui alimentent la queue d'entrée et vident les queues de sortie. Pour chaque type de transfert, il existe un programme background : lecteur → disque, disque → imprimante, disque → perforateur, disque → bande, bande → disque, etc. Un certain nombre de ces programmes sont fournis par le constructeur, d'autres peuvent être ajoutés au niveau de l'exploitation.

Le sous-système permet de travailler avec des fichiers permanents sur disque.

Le programme 63 est chargé essentiellement de traiter les requêtes adressées par tous les autres programmes, ainsi que les communications avec l'opérateur. Il se compose d'une partie résidente et de plusieurs overlays.

Le tout est dirigé par INC (interrupt control) qui gère le système d'interruption.

Le système d'exploitation comporte un certain nombre de tables qui contiennent des renseignements sur les différents programmes, les opérations d'entrée-sortie, etc.

A tout instant, elles reflètent l'état du système, tant physique que logique.

L'ensemble des programmes est contrôlé par la table PSL (program status list), de longueur fixe : 64 mots, 1 par programme. Des bits décrivent l'état du programme (présent, initialisé, en cours d'exécution, en attente, alerté par une interruption, en perdition, etc.). Ce mot contient en outre un pointeur vers sa table PCT.

Pour chaque programme présent, il existe une table PCT (program control table) de 11 mots, contenant essentiellement l'état des registres et un pointeur vers la table LUB (logical unit block) correspondante (voir section 3.4).

INC, appelé exclusivement par le système d'interruption, ne possède pas de table. C'est également INC qui décide à quel programme le contrôle doit être donné.

Tous ces programmes, à part INC, certaines parties des programmes 2 et 63 sont exécutés avec le système d'interruption actif (c'est-à-dire peuvent être interrompus à tout moment).

Lors d'une interruption, le contrôle est transmis à l'adresse (0)1 dans INC qui analyse les causes de l'interruption, les efface, puis transmet le contrôle à celui des programmes qui le demande, dont le niveau est le plus élevé. Cela peut être :

- le programme 63 dans le cas d'une intervention de l'opérateur ou d'une requête (voir section 3.3) en provenance du groupe du programme central, d'un programme background ou du programme 63 lui-même;

- un programme background s'il s'agit d'une interruption le concernant;
- le programme de niveau 1 s'il s'agit d'une interruption relative au programme central (voir section 3.3).

3.3 Requêtes et unités logiques

Les programmes des différents niveaux ont la possibilité d'effectuer des requêtes (dans la terminologie IBM : service call) au système. Ces requêtes sont introduites dans le langage assembleur sous forme de macro-instructions; elles commencent toutes par une instruction de branchement à l'adresse (0)0, c'est-à-dire dans la zone de mémoire protégée, ce qui provoque une interruption et donne le contrôle à INC.

Toutes les requêtes ne sont pas accessibles à n'importe quel programme : certaines ne le sont qu'au programme central, d'autres qu'aux programmes background, d'autres à tous les programmes. Il existe, par ailleurs, des requêtes à l'usage exclusif du système. Pour celles-ci, le programme 2 a les mêmes privilèges que le programme 63.

Certaines requêtes sont traitées directement par INC, dans la plupart des cas c'est le programme 63 qui reçoit le contrôle : il recherche et analyse les paramètres puis traite la requête. Sommairement, les requêtes peuvent être classées en trois groupes :

- opérations d'entrée-sortie;
- contrôle des interruptions internes;
- demande de renseignements ou de services divers au système.

Les requêtes d'entrée-sortie sont les plus complexes; leur traitement, lié à celui du système d'interruption, est réparti entre le programme 63 et INC. L'étude détaillée de ce processus fait l'objet des sections 3.5 à 3.7. .

Pour toutes les opérations d'entrée-sortie, le programmeur travaille avec la notion d'unité logique. Ce concept n'est

pas soumis aux mêmes règles dans le programme central et dans les programmes background.

Dans le groupe du programme central, les unités dites du programmeur portent des numéros quelconques entre 1 et 49. Les numéros 50 à 79 ont, par contre, des significations bien précises.

Dans les autres programmes, les unités doivent être numérotées en séquence, à partir de 1.

Le but essentiel des requêtes d'entrée-sortie est de décharger le programmeur du traitement des interruptions et de lui permettre de travailler avec autant d'efficacité que possible.

Les requêtes qui provoquent un transfert de données ou une autre action mécanique ne font que lancer l'opération, le signal de la fin d'opération étant marqué par une interruption.

Les paramètres d'une requête d'entrée-sortie sont :

- u : numéro de l'unité logique;
- cwa : adresse du premier mot de contrôle de la chaîne (seulement en cas de transfert de données);
- ra : adresse à laquelle le contrôle est transféré en cas de rejet de la requête;
- ia : adresse de la routine d'interruption à laquelle le contrôle doit être transféré à la fin de l'opération (cette indication peut être omise).

Ces paramètres peuvent être indicés ou non.

Quelques requêtes :

READ	}	u, cwa, ra, ia	lecture
WRITE			écriture
MERKEF	}	u, ra, ia	écriture d'une fin de fichier
REWIND			rebobinage
STATUS	u		consultation de la table donnant l'état physique et logique de l'unité
DYSTAT	u		analogue à STATUS, avec en plus la consultation de l'état physique de l'unité
MODE		u, ra, d ₁ , d ₂ , d ₃ , d ₄	déclaration des modes de fonctionnement de l'unité

d₁, d₂, d₃ et d₄ peuvent être entre autres :

BY (bypass)	toutes les requêtes, sauf STATUS, MODE et RELEASE sont sans effet
RO (read only)	seules les opérations de lecture sont permises
WO (write only)	seules les opérations d'écriture sont permises
RW (read and write)	toutes les opérations physiquement possibles sont permises
LO (low density)	densité 200 caract./pouce
HI (high density)	densité 556 caract./pouce
HY (hyper density)	densité 800 caract./pouce
BIN	fonctionnement en mode binaire (c'est-à-dire non codé)
BCD	fonctionnement en mode BCD (binaire codé décimal)

Les modes BIN et BCD peuvent correspondre à une réalité hardware (bandes, cartes) ou software (voir section 3.8).

Remarques :

- 1) La première opération sur une unité est nécessairement précédée de l'assignation, dont le but est essentiellement d'effectuer la correspondance entre une unité logique et une unité physique; à cette occasion un certain nombre d'opérations sont effectuées sur l'unité.

Dans le cas du programme central, l'assignation est effectuée sans autre par le programme 63, selon les indications figurant sur les cartes de contrôle.

Exemples :

⁷₉EQUIP, 7 = MT

⁷₉EQUIP, 9 = CP

Les unités sont désignées ici par des mnémoniques connus du système. Une unité de bande magnétique est assignée à l'unité 7, et un perforateur à l'unité 9. Dans le cas de bande magnétique, la carte de contrôle peut également porter l'indication d'une étiquette.

En l'absence de toute déclaration, une unité logique est assignée à une portion de disque.

Dans un programme background, l'assignation doit faire l'objet d'une requête.

Exemples :

```
ASSIGN      1,CP
```

```
ASSIGN      2,MT4
```

Un perforateur est assigné à l'unité 1, l'unité de bande numéro 4 à l'unité logique 2.

- 2) L'opération inverse est la libération (release). Elle peut être demandée dans un programme central ou background par la requête

```
RELEASE     u,ra
```

Le traitement de cette requête provoque également l'exécution d'un certain nombre d'autres opérations sur l'unité. En fin de travail, le programme 63 libère toutes les unités encore assignées au programme.

- 3) Au moment où la requête est complètement exécutée, le système donne le contrôle à l'adresse d'interruption (pour autant qu'elle ait été définie) pour l'exécution d'une routine d'interruption fournie par le programme qui a effectué la requête. La fin de l'exécution d'une requête est généralement marquée par une ou plusieurs interruptions, mais ce n'est pas toujours le cas. Ainsi, rien ne se passe sur une unité de bande si une requête REWIND est faite alors que la bande est déjà rebobinée, mais la routine d'interruption est tout de même exécutée. L'expression "routine de fin d'opération" serait donc plus correcte.
- 4) Dans le programme central, les requêtes contenant une adresse d'interruption sont, en principe, effectuées au niveau 0. Pour l'exécution de la routine d'interruption, il y a passage au niveau 1, c'est-à-dire dans un état quelque peu différent et privilégié. Ce changement de niveau n'a lieu que dans le programme central; dans un programme background, tout se

passé au même niveau.

- 5) La fin d'une routine d'interruption est marquée par une requête RETURN. Dans le programme central, cela provoque un retour du niveau 1 au niveau 0, là où celui-ci avait été interrompu. En effectuant une requête RETURN, un programme background indique au système qu'il abandonne momentanément le contrôle (en général, il vient d'initialiser une nouvelle opération d'entrée-sortie et il attend une autre interruption). Le contrôle est alors transmis à un autre programme background de niveau moins élevé, ou au groupe du programme central.
- 6) A côté des numéros logiques externes, donnés par le programmeur aux unités du programme central, le système définit les numéros logiques internes, en séquence à partir de 1. La correspondance entre numéros logiques externes et internes est donnée par une table appelée LEQ (logical equivalence table). Cette table n'existe que pour le programme central, puisque dans les autres programmes, les unités sont déjà numérotées à partir de 1.
- En associant à chaque unité logique son LUD (logical unit designate), le système définit une notion valable à tous les niveaux de programme : elle réunit le numéro logique et sa provenance dans un même élément.

$$\text{LUD} = \begin{cases} \text{numéro logique interne pour une unité du groupe} \\ \text{du programme central (0 < LUD < 400}_8\text{);} \\ 100_8 \cdot \text{numéro du programme} + \text{numéro logique à} \\ \text{l'intérieur de celui-ci dans un autre programme.} \end{cases}$$

3.4 Tables

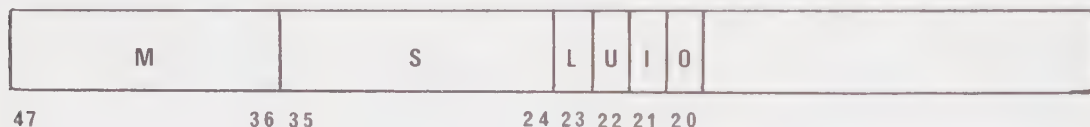
Les tables jouent un rôle primordial dans le traitement des requêtes d'entrée-sortie. Certaines servent à décrire la configuration, d'autres sont attachées aux programmes. Les tables sont composées de mots divisés en champs de longueur variable. Par convention de notation, le champ CH de

la table XYZ est noté XYZ.CH, .XYZ désigne la longueur de la table XYZ.

Dans la description qui suit, seuls les champs les plus importants sont mentionnés. Les disques jouent un rôle privilégié et ont un traitement tout à fait spécial, auquel sont attachés plusieurs champs. Ceux-ci sont également omis. Pour les champs d'un seul bit, la condition énoncée correspond à la valeur 1 de ce bit.

HTL (hardware type list)

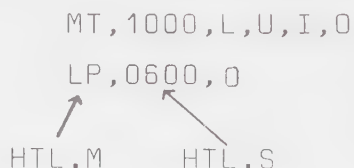
Dans cette table figure une entrée d'un mot pour chaque type d'équipement.



- HTL.M code mnémonique de 2 caractères, désignant le type d'équipement;
- HTL.S masque donnant tous les bits possibles du statut de l'équipement marquant la fin d'une opération avant l'interruption du canal signalant la fin du transfert des données. Le cas typique est une condition anormale détectée sur l'unité. L'utilisation de ce champ est décrite dans la section 3.6.
- HTL.L les unités sont susceptibles de porter une étiquette;
- HTL.U les unités peuvent être déchargées;
- HTL.I les unités sont capables de lire;
- HTL.O les unités sont capables d'écrire.

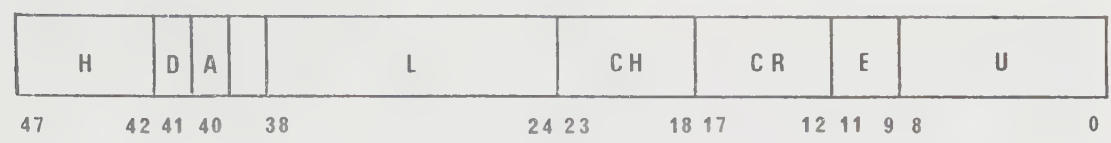
La table HTL est construite par PREDRUM à partir de la forme symbolique externe lue sur la bande système. Les champs décrits ci-dessus sont constants.

La forme symbolique des entrées relatives aux bandes magnétiques et à l'imprimante se présente ainsi :



AET (available equipment table)

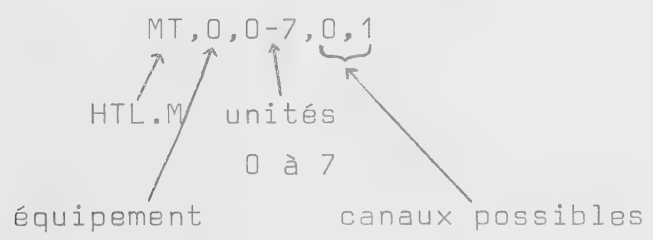
Il existe une entrée AET d'un mot pour chaque unité physique.



- AET.H ordinal du type d'unité, correspond à l'ordinal dans HTL;
- AET.D l'unité est hors service;
- AET.A l'unité est présentement assignée;
- AET.L contient la valeur du LUD lorsqu'une interruption est attendue, vaut 0 sinon;
- AET.CH contient le numéro du canal après la connection et 40₈ en l'absence de celle-ci;
- AET.CR ordinal se rapportant à la table CRLIST;
- AET.E numéro d'équipement;
- AET.U numéro d'unité.

La table AET est construite par PREDRUM à partir de la forme symbolique externe lue sur la bande système. Une seule image carte sur la bande peut donner lieu à plusieurs entrées dans la table. Les trois derniers champs, ainsi que AET.H dépendent uniquement de la configuration et sont constants. Les autres champs varient et reflètent l'état de l'unité. AET.E et AET.U constituent une partie du code de connection utilisé par l'instruction CONN.

La forme symbolique des entrées AET relatives aux bandes magnétiques se présente ainsi (voir configuration, section 2.4) :



CRLIST (controller list)

Cette table indique par quels canaux un équipement est connectable, chaque entrée est formée de 8 champs de 6 bits, contenant chacun un numéro de canal ou 40, lorsqu'il n'y a pas d'autre canal. L'information extraite de l'un de ces champs est utilisée par les instructions machine d'entrée-sortie. La construction des tables AET et CRLIST implique qu'un équipement pouvant être atteint par deux canaux différents doit porter le même numéro pour chacun d'eux, ce qui n'est pas une obligation pour le hardware.

CRLIST est construite par PREDRUM à partir des informations lues sur les images carte donnant la table AET sous sa forme symbolique. La longueur de CRLIST est au plus égale à celle de HTL.

CHTAB (channel table)

Chaque canal a un mot dans cette table



CHTAB.A le canal effectue un transfert d'information;

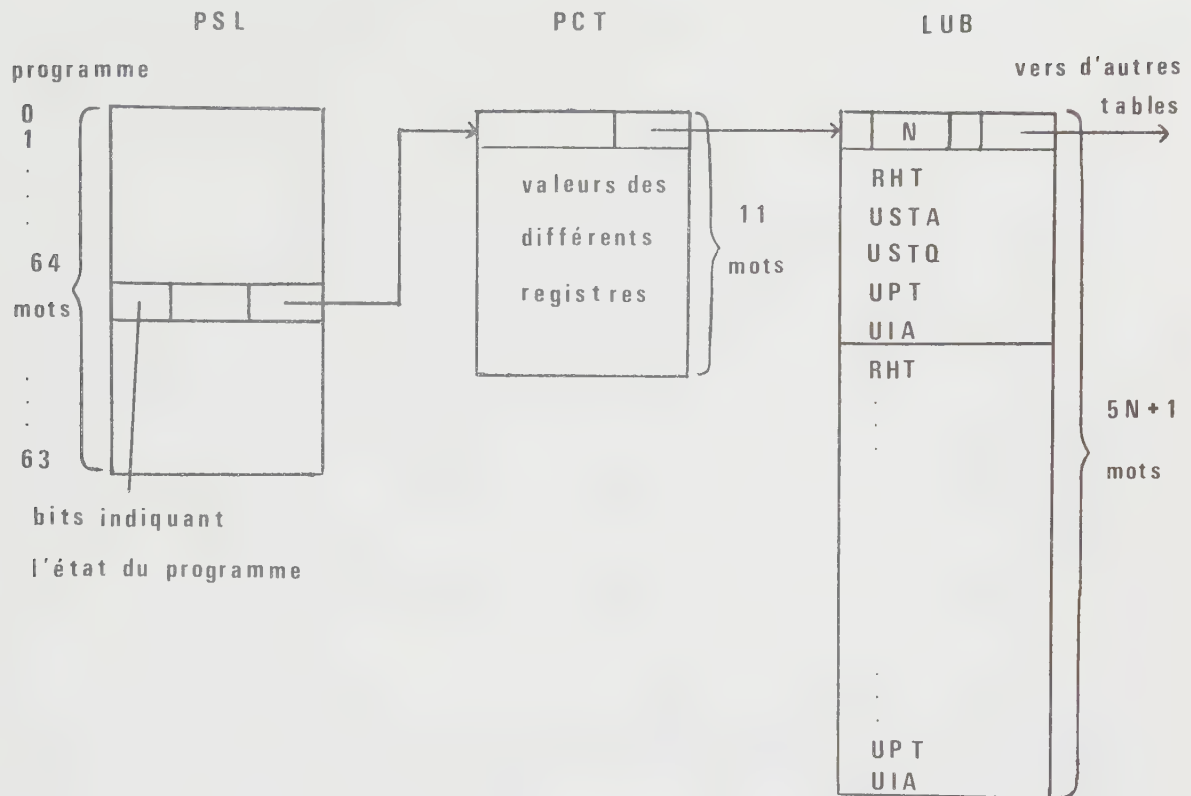
CHTAB.K contient l'ordinal AET de l'unité présentement connectée, et 0 en l'absence de connection.

La place pour CHTAB est réservée par PREDRUM. Cette table reflète constamment l'état des canaux.

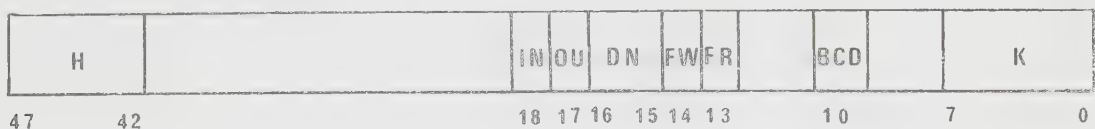
Les autres tables sont liées aux unités logiques. Chaque programme dispose d'un LUB (logical unit block) dont l'adresse est donnée par son PCT. Le premier mot de LUB contient le nombre d'unités logiques N du programme et un pointeur vers d'autres tables.

Alors que PCT est de longueur fixe, LUB varie d'un programme à l'autre, sa longueur vaut $5N + 1$. Chaque unité logique dispose de 5 mots appelés RHT, USTA, USTQ, UPT, UIA.

Les tables sont donc chaînées comme suit :



RHT (running hardware table)



- RHT.H ordinal du type d'unité, correspond à l'ordinal dans HTL;
- RHT.IN la dernière opération est une lecture;
- RHT.OU la dernière opération est une écriture;
- RHT.DN indicateur de densité (0 : aucune déclaration, 1,2,3 : 200, 556, 800 caractères/pouce respectivement);
- RHT.FW les opérations d'écriture sont interdites;
- RHT.FR les opérations de lecture sont interdites (il s'agit là de restrictions logiques, tandis que HTL indique les possibilités physiques);
- RHT.BCD l'unité travaille en mode BCD;
- RHT.K ordinal dans AET de l'unité physique assignée, 0 avant l'assignation ou après la libération.

Remarque :

Les champs RHT.DN, RHT.FW, RHT.FR et RHT.BCD peuvent être fixés par la requête MODE.

USTA (unit status table A)

USTA contient le dernier mot de contrôle utilisé ou en cours d'utilisation. La description des différents champs correspond exactement à celle de la section 2.2.



USTA est remis à 0 par une requête RELEASE.

USTQ (unit status table Q)



- USTQ.A l'unité est active (voir section 3.5);
- USTQ.B le sens de ce bit dépend de la valeur de USTQ.A, il est précisé dans la section 3.5;
- USTQ.T l'unité n'est ni une bande, ni une unité bloquée (disque);
- USTQ.Y l'unité fait l'objet d'un bypass;
- USTQ.SRB (status reply bits) ces 12 bits correspondent plus ou moins au statut physique de l'unité donné par l'instruction COPY. Les bits 32 (USTQ.RDY) et 33 (USTQ.BSY) correspondent respectivement aux bits 0 et 1 fournis par cette instruction (prêt, occupé). Les bits 35-37, 42, 43 ont une signification globale, applicable à toutes les unités.
- USTQ.EOF (bit 35) une fin de fichier a été lue ou écrite lors de la dernière opération;

USTQ.PAR (bit 42) une erreur de parité a eu lieu au cours de la dernière opération.

Les bits 34, 38-41 dépendent exclusivement du type d'équipement. Dans la plupart des cas, les bits donnés par l'instruction COPY sont permutés afin de respecter les conventions ci-dessus.

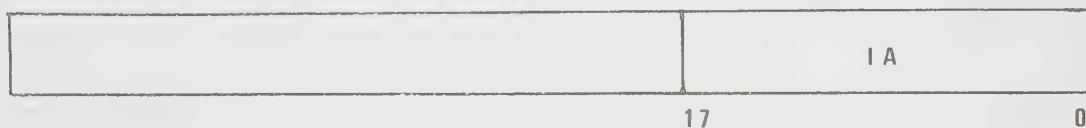
USTQ.F contient un code de fonction (voir section 3.5);

USTQ.RH contient l'ordinal dans HTL si USTQ.T = 1;

USTQ.CWA adresse du mot de contrôle suivant.

A l'entrée d'une routine d'interruption ou en réponse à une requête STATUS ou DYSTAT, les registres A et Q contiennent USTA et USTQ respectivement (USTQ.F est toutefois remplacé par le numéro logique).

UIA (unit interrupt address)



Ce mot contient l'adresse de la routine d'interruption spécifiée dans la requête.

UPT (unit pointer table)

Ce mot n'a une signification fixe que pour les unités bloquées; dans les autres cas, il peut contenir des informations complémentaires quelconques, selon les besoins du driver et les particularités de l'équipement.

Lors du traitement d'une requête d'entrée-sortie, un certain nombre de pointeurs, liés aux tables des unités logiques, jouent un rôle très important. Ce traitement étant réparti entre le niveau 63 et INC, tous ces pointeurs sont à double; ceux commençant par la lettre I se rapportent à INC (système d'interruption inactif), ceux commençant par la lettre A se rapportent au programme 63 (système d'interruption actif).

Les 5 entrées relatives à une unité logique sont atteintes par adressage indirect, à travers les pointeurs :

IRHT, IUSTA, IUSTQ, IUPT, IUIA
ARHT, AUSTA, AUSTQ, AUPT, AUIA

Exemple :

DLDA AUSTA,7 requête STATUS, telle qu'elle est
 traitée par le programme 63.

D'autres cellules contiennent des valeurs et non pas des adresses :

IORD, AORD contiennent l'ordinal AET (=RHT.K);
ILUD, ALUD contiennent le LUD;
INTORD contient l'ordinal AET de l'unité qui
 a provoqué la dernière interruption.

Dans tout système d'exploitation, les tables et les pointeurs sont des éléments essentiels et doivent par conséquent être protégés. Dans le système DISK SCOPE, ils sont situés dans la portion inférieure de la mémoire et sont protégés par le "bounds register". Seuls INC, le programme 63 et, dans quelques cas, l'exécutif peuvent les modifier.

L'intérêt des tables est que les appels aux différentes routines ne sont pas alourdis par des listes d'arguments : à l'exception d'un ou deux éléments transmis par les registres, toutes les autres informations nécessaires sont déjà placées à des endroits déterminés.

Les informations relatives aux unités physiques et aux unités logiques sont bien séparées. Les deux éléments essentiels sont le LUD et IORD, l'ordinal AET. Le premier permet le passage de l'unité physique à l'unité logique puisqu'il est stocké dans le champ AET.L, le second permet le passage inverse, il se trouve dans le champ RHT.K.

Il convient de remarquer que le champ AET.L peut changer de valeur d'une opération à l'autre car il est possible que des unités logiques différentes utilisent successivement la même unité physique (p. ex.: la machine à écrire de la console ou les disques).

Pour les tables constituant le LUB, l'adressage indirect est très avantageux. Lors de chaque opération d'entrée-sortie, les pointeurs sont mis à jour par la routine IOSET, les registres d'indices restent ainsi disponibles pour toutes les autres opérations.

Enfin, cette conception des tables permet sans autre la réentrance. En particulier, INC, détectant la fin d'une opération sur une unité peut interrompre le programme 63 en train de traiter une requête sur une autre unité.

C'est grâce au double jeu des pointeurs, I et A, que cette possibilité existe. La routine IOSET, qui peut être appelée aussi bien par le programme 63 que par INC, est elle-même réentrante.

3.5 Traitement d'une requête d'entrée-sortie au niveau 63 et dans INC

Toutes les requêtes commencent par un branchement à l'adresse (0)0, cette instruction contient en outre un code d'appel, identifiant la requête (voir annexe 1). Ce branchement produit une interruption à cause de la protection de mémoire (bounds fault) et INC prend le contrôle. Après effacement de l'interruption, l'instruction responsable de celle-ci est analysée et reconnue comme faisant partie d'une requête. INC vérifie ensuite si la requête est légale dans le programme qui l'a faite; si ce n'est pas le cas, celui-ci est abandonné. Pour certaines requêtes privilégiées, INC se charge immédiatement du traitement, mais dans la plupart des cas, le contrôle parvient à la routine IPASS2 qui détermine le programme qui doit le recevoir à son tour. En l'occurrence, c'est le programme 63; il reçoit le contrôle par une instruction de transfert placée à l'adresse (0)0, ce qui remet le système d'interruption actif. Le traitement des requêtes a deux entrées possibles : REQUEST et ISENTRY. La première entrée correspond aux requêtes effectuées dans les niveaux 0 à 62 et la seconde à celles effectuées par le niveau 63 lui-même : il est possible,

en effet, que l'exécution d'une requête donne lieu à d'autres requêtes imbriquées. Le programme 63 dispose pour cela d'une pile permettant la récursivité. Après un certain nombre d'opérations, appartenant en propre à chacune des deux entrées, les deux traitements se rejoignent.

Dans la partie commune à toutes les requêtes s'effectuent l'extraction des paramètres, la réduction des adresses indicées (dans les paramètres) en adresses absolues, le stockage de ces valeurs dans une zone réservée, conformément aux spécifications figurant dans la table STV (system transfer vector) qui précise la forme de la requête et le sens des paramètres. Un appel à IOSET met à jour les cellules de type A, s'il s'agit d'une entrée-sortie.

Chaque requête a ensuite sa propre routine de traitement qui porte le même nom que la requête.

Le traitement des requêtes d'entrée-sortie se compose de deux phases distinctes : le contrôle et l'exécution.

C'est la routine IOC (input-output control) qui effectue la première phase, elle peut être appelée avec 17 paramètres différents. A chacun d'eux correspond une vérification ou une action ainsi qu'un bit dans la séquence d'appel. Chaque requête en sélectionne un certain nombre lors de l'appel à IOC; les paramètres transmis par le programme ayant effectué la requête doivent être en accord avec les caractéristiques de celle-ci, toute erreur provoque l'abandon dudit programme.

La deuxième phase commence par un appel à la routine IOEX, cet appel marque le début de l'exécution proprement dite de l'opération. C'est à ce niveau qu'intervient une nouvelle différenciation dans le traitement : il dépend également du type d'équipement. Un programme, appelé driver, fait le lien entre IOEX et l'équipement. Pour chaque type d'équipement, il existe un driver, par convention il porte le nom DRXXD, où XX désigne le mnémonique contenu dans HTL.M. C'est sous le contrôle du driver que sont exécutées les instructions machine d'entrée-sortie.

Les adresses des drivers sont contenues dans la table DEP (driver entry point) parallèle à HTL. Certains drivers résident en permanence en mémoire, d'autres sont chargés sur demande à partir du disque. C'est surtout la fréquence d'utilisation du type d'équipement qui détermine si un driver doit être déclaré (sur la bande système) résident ou non. IOEX a pour paramètre un code d'opération à effectuer par le driver.

code	signification	requête associée
RDF	lecture	READ
WRF	écriture	WRITE
STA	consultation du statut	DYSTAT
REL	libération	RELEASE
REW	rebobinage	REWIND
WEF	écriture d'une fin de fichier	MARKEF
SDH	mise en densité HI	--
RDY	sélection d'interruption sur unité disponible	READY

Remarque :

Dans certains cas, IOC peut déjà faire un ou plusieurs appels à IOEX. Un cas typique est l'assignation, au cours de laquelle IOC fait plusieurs appels au driver avec les codes STA et SDH entre autres. Donc, à une requête peuvent très bien correspondre plusieurs entrées dans le driver. Réciproquement, deux requêtes différentes peuvent appeler le driver avec un même code.

IOEX place le code reçu dans le champ USTQ.F de l'unité, charge dans le registre A le contenu de ALUD et effectue une requête STARTUP, qui est une des requêtes traitées par INC. Le traitement passe ainsi du niveau 63 à INC, il ne peut donc plus être interrompu.

STARTUP appelle IOSET pour mettre à jour les pointeurs de type I, met à 1 le bit USTQ.A et appelle DRIVERU.

La routine d'appel aux drivers s'appelle DRIVER; elle possède deux entrées : DRIVERU, pour l'initialisation des opérations, DRIVERL, pour les appels au moment des interruptions.

De même, chaque driver a deux entrées contenues dans chacune des deux demi-instructions de son premier mot : DRIVERU donne le contrôle à la demi-instruction supérieure, DRIVERL à la demi-instruction inférieure. Que le driver soit appelé par DRIVERU ou par DRIVERL, il doit disposer des mêmes instructions d'entrée-sortie et des mêmes pointeurs mis à jour. Le traitement d'une interruption pendant l'exécution d'un driver pourrait modifier ces valeurs, c'est pour cette raison que les drivers font partie de INC et que lors d'une initialisation, la requête STARTUP permet de quitter le niveau 63 pour INC.

DRIVER détermine le driver à appeler, vérifie s'il est en mémoire, lui donne le contrôle s'il est présent ou, sinon, met l'opération en attente.

Le comportement à l'intérieur des drivers fait l'objet des deux sections suivantes. A la sortie de celui-ci, et si l'opération a pu être initialisée, le contrôle retourne généralement à DRIVER, puis à INTER3 (dans INC) qui traite alors d'éventuelles interruptions, donne le contrôle à IPASS2 qui le rend au programme 63, c'est-à-dire à la routine de traitement de la requête. Celle-ci peut encore accomplir un certain nombre d'opérations avant d'effectuer la requête RETURN, traitée elle aussi par INC, et qui aboutit tôt ou tard à IPASS2.

Cette routine donne alors le contrôle à celui des programmes le demandant qui a le niveau le plus élevé. Cela peut être le programme qui a effectué la requête ou un autre.

Après un laps de temps plus ou moins long, la ou les interruption(s) se produisent. INTER6, dans INC, détermine tout d'abord le canal dont provient une telle interruption puis appelle la routine ICHAN, qui exécute l'instruction CPJ afin de préciser la provenance (canal ou numéro d'équipement) et appelle DRIVERL. Cette routine appelle IOSET pour la mise à jour des pointeurs I et donne le contrôle à l'entrée inférieure du driver. Celui-ci peut rendre le contrôle temporairement s'il attend encore d'autres interruptions, ou définitivement à DRFIN si l'opération est entièrement terminée.

C'est dans DRFIN que le bit USTQ.A est remis à 0, tandis que les bits USTQ.B et PSL.A prennent la valeur 1 si la requête

a mentionné une adresse d'interruption. Le contrôle passe alors de DRFIN à DRIVER, à ICHAN, à INTER6 et finalement à IPASS2. La présence du bit PSL.A signale à IPASS2 qu'une routine d'interruption doit être exécutée, l'adresse d'interruption est alors extraite de UIA, stockée à l'adresse (0)0 et l'entrée dans la routine d'interruption par l'adresse (0)0 remet le système d'interruption actif.

Cette description du traitement d'une requête est très linéaire et simplifiée. En fait des éléments étrangers à ce traitement peuvent intervenir :

- Interruptions provenant d'opérations d'entrée-sortie initialisées plus tôt. Dans ce cas, le programme 63 est interrompu, INC ne l'est pas, il s'occupe des interruptions pouvant survenir, lors du passage suivant à INTER6.
- Avant d'entrer dans IPASS2, INC exécute PRS (process stack) et cherche à satisfaire les demandes que les drivers ont pu mettre en attente.

Quelques points laissés en suspens dans la section 3.4 s'expliquent maintenant :

- 1) USTQ.A correspond à une activité logique de l'unité, tandis que USTQ.BSY signifie une activité physique. Si USTQ.BSY = 1, alors USTQ.A = 1, le contraire n'est pas toujours vrai.
- 2) Le bit USTQ.B n'est utilisé qu'en fin d'opération, avant cela, le driver peut en disposer. En fin d'opération, il signifie qu'une routine d'interruption relative à l'unité doit être exécutée; tant qu'il vaut 1, aucune nouvelle opération ne peut être initialisée sur l'unité en question.
- 3) La requête STATUS place dans A et Q le contenu de USTA et de USTQ, tandis que la requête DYSTAT fait un appel au driver avec le code de fonction STA. Toutefois, si USTQ.A = 1, DYSTAT se comporte comme STATUS.

3.6 Entrées et sorties d'un driver, routines à disposition des drivers

Une unité physique est toujours dans l'un des états logiques suivants :

état	caractéristiques
- connectée	AET.CH contient le numéro du canal;
- occupée	l'unité attend une interruption, AET.L contient le LUD;
- occupée avec canal actif	un transfert est en cours, CHTAB.A = 1;
- libérée	AET.CH = 40 ₈ , AET.L = 0. Cette libé- ration ne concerne que les tables décri- vant l'état des périphériques, elle n'a rien à voir avec la libération provoquée par la requête RELEASE.

Les drivers ont à leur disposition un certain nombre de rou-
tines résidentes, qui effectuent la mise à jour des tables et
des instructions machine.

Celles-ci se trouvent dans des mots appelés CONN, EXTF, (fonc-
tion sur un équipement), CEXTF (fonction sur un canal), BEGR,
BEGW, COPY (statut du canal et de l'équipement) et COPYS (sta-
tut de l'équipement seulement).

Elles sont mises à jour à l'aide des valeurs de AET.E, de
AET.U et de CRLIST par la routine XCONS.

Les routines contrôlant les entrées-sorties sont les suivantes :

XCONN

Séquence d'appel : LDA IORD
 RTJ XCONN

Au retour : (A) ≥ 0 : l'unité est connectée (logiquement et
physiquement);

(A) = -0 : l'unité n'est pas connectée car aucun
chemin libre n'a été trouvé;

(A) = -1 : la connection a été physiquement rejetée.

"Aucun chemin libre n'a été trouvé" signifie :

- l'unité est déjà occupée;
- une autre unité est déjà connectée par le même canal et le même équipement;
- le canal choisi est déjà actif.

Si $(A) \geq 0$ à la sortie de XCONN, AET.CH contient le numéro x du canal et CHTAB (x) la valeur de IORD.

Il est indispensable que lors de l'appel $1 \leq (A) \leq .AET$, sinon le système se trouve dans une situation anormale (DISASTER) qui nécessite une intervention de l'opérateur.

RCONN

Cette routine est analogue à XCONN, mais en plus exécute l'instruction COPYS et au retour le contenu de A est différent :

- (A) = 1 : l'unité est connectée et prête;
- (A) = -0 : l'unité est connectée mais pas prête;
- (A) = -1 : l'unité n'est pas connectée.

RECONN

Cette routine ressemble également à XCONN, la séquence d'appel est la même. Toutefois :

- la connection est possible si l'unité est déjà occupée (RECONN est utilisé au moment de l'arrivée des interruptions);
- le rejet de la connection n'est pas admis ici et provoque un appel à DISASTER.

Au retour, $(A) \geq 0$ signifie : unité connectée.

SEXTF

Séquence d'appel : ENA c
 RTJ SEXTF

SEXTF stocke le code de la fonction c dans l'instruction EXTF, l'exécute et retourne avec $(A) = -0$ si elle a été rejetée et avec $(A) = +0$ sinon.

XEXTF

Séquence d'appel : ENA c
 RTJ XEXTF

XEXTF appelle SEXTF jusqu'à ce que la fonction soit acceptée et place la valeur de ILUD dans AET.L. L'unité devient donc occupée après un appel à XEXTF.

SBEGR

Séquence d'appel : ENA cwa
 RTJ SBEGR

L'adresse cwa du mot de contrôle est placée dans l'instruction BEGR, qui est ensuite exécutée. Au retour (A) = +0 en cas de succès, le rejet n'est pas admis et donne le contrôle à DISASTER.

XBEGR

Séquence d'appel : ENA cwa
 RTJ XBEGR

Cette routine exécute la fonction 12₈ sur le canal x (sélection d'interruption sur fin de chaîne et sur erreur) appelle SBEGR, et place la valeur de ILUD dans AET.L et 1 dans CHTAB.A(x). L'unité devient ainsi occupée avec canal actif.

SBEGW et XBEGW sont rigoureusement analogues à SBEGR et à XBEGR, mais font les transferts dans l'autre sens.

COPY et COPYS servent à prendre le statut, ils sont exécutés à l'aide de l'instruction EXEC (voir annexe 1).

XREL

Séquence d'appel : ENA c
 RTJ XREL

Cette routine remet à 0 AET.L, CHTAB (x), place la valeur 40₈ dans AET.CH, appelle SEXTF avec le code c et boucle en cas de rejet. L'unité physique est ainsi libérée.

Remarques :

- 1) Il est tout à fait possible qu'une unité soit logiquement occupée sans être physiquement connectée. En effet, soit u_1 une unité attendant une interruption sans qu'il n'y ait de transfert de données. Pendant ce temps, une unité u_2 , rattachée à un autre équipement sur le même canal, peut se connecter à lui. La ligne d'interruption fonctionnant, que la connection physique soit présente ou non, l'unité u_1 n'est pas perturbée.
- 2) Après un appel à SBEGR (ou SBEGW) le canal devient actif, logiquement cependant, l'unité reste dans l'état où elle se trouvait auparavant (connectée seulement).

FETCW

Séquence d'appel : ENQ f
 RTJ FETCW

FETCW va chercher le mot de contrôle situé à l'adresse USTQ.CWA et le place dans USTA. USTQ.CWA est augmenté d'une unité.

Les mots de contrôle du type IOJP sont exécutés, un mot de contrôle chaîné, dont le compte de mots est nul, est ignoré et un passage au mot de contrôle suivant a lieu.

Une chaîne trop longue ou bouclant sur elle-même, de même qu'un mot de contrôle désignant une portion de mémoire inexistante sont considérés comme des erreurs.

Si $f \neq 0$ FETCW vérifie que le mot de contrôle respecte la protection de mémoire, si $f = 0$ ce test n'est pas fait. En principe $f \neq 0$ en cas de lecture seulement.

Au retour, $(Q) \neq 0$ signifie qu'une erreur a été détectée.

MOV CW

Séquence d'appel : ENQ f
 RTJ MOV CW

MOV CW effectue autant d'appels à FETCW qu'il est nécessaire pour amener, dans une zone réservée au canal, toute une chaîne de mots de contrôle. Chaque canal dispose d'une zone de .CWORDS (paramètre d'installation du système) mots, une erreur

est donc détectée si la chaîne comporte plus de .CWORDS mots de contrôle effectifs.

La raison de ce transfert dans une zone réservée est la suivante : le système doit être totalement protégé. Si l'opération se déroulait sous le contrôle d'une chaîne située dans le programme, celui-ci pourrait modifier, volontairement ou involontairement, un des mots de la chaîne pendant l'exécution du premier; ainsi une chaîne, même testée avant l'opération, pourrait détruire le système.

Le retour de MOV CW est identique à celui de FET CW.

La principale routine, intervenant lors de l'arrivée des interruptions externes, s'appelle ICHAN. Elle fait partie de INC qui l'appelle par INTER6 avec le numéro x du canal dans B1.

ICHAN appelle tout d'abord XCONS, puis exécute l'instruction CPJ sur le canal x. La suite dépend des diverses réponses que peut donner cette instruction.

- 1) En cas d'erreur (parité au niveau du canal ou dans le mot de contrôle), la routine de récupération IOERR est appelée. Ce cas est exceptionnel.
- 2) En cas d'interruption sur fin de chaîne (fin normale après l'exécution d'une chaîne, cette interruption a été sélectionnée par XBEGR ou XBEGW), cette interruption est effacée, CHTAB.A(x) est remis à 0. Après un passage dans RECONN, le driver est appelé par DRIVERL avec (Q) = +0. Au retour, le contrôle est redonné à ICHAN.
- 3) En cas d'interruption en provenance d'un équipement, trois possibilités peuvent se présenter :
 - 3a) CHTAB.A (x) $\neq$ 0, le canal est actif, mais pour le compte d'un autre équipement;
 - 3b) CHTAB.A (x) $\neq$ 0, le canal est actif pour le compte de l'équipement qui a provoqué l'interruption;
 - 3c) CHTAB.A (x) = 0, le canal est inactif.

Dans le cas 3a), ICHAN boucle en attendant que l'équipement utilisant le canal ait terminé son transfert. L'interruption du canal doit être traitée en priorité et il serait de toute

façon impossible d'effectuer une connection ou une fonction tant que le canal est actif.

Dans le cas 3b), l'équipement a envoyé une interruption pendant qu'il était en train de travailler avec le canal. Il s'agit d'une condition spéciale ou anormale.

ICHAN effectue alors l'instruction COPYS et regarde si l'un des bits figurant dans le masque HTL.S est à 1. Si ce n'est pas le cas, ICHAN boucle en attendant l'interruption du canal, comme dans le cas 3a). Sinon, l'interruption de l'équipement est traitée comme une interruption du canal, après l'exécution de la fonction destinée à arrêter l'activité de celui-ci. Un cas typique est la lecture d'une fin de fichier sur une bande magnétique : le compte de mots n'est pas épuisé, le canal est encore actif et pourtant l'opération doit être considérée comme terminée.

Dans le cas 3c), l'interruption a été sélectionnée par une fonction à l'équipement. La table AET est parcourue pour trouver une unité occupée sur l'équipement et le canal envisagés (champs AET.CH et AET.E). Si la recherche est infructueuse, le contrôle est donné à DISASTER, sinon RECONN est appelé, puis le driver par DRIVERL avec (Q) = -0.

Après le traitement des cas 1), 2) ou 3), le contrôle revient toujours à l'instruction CPJ.

4) Il n'y a plus d'interruption, le contrôle retourne à INTER6.

Lorsqu'ils ont terminé leur travail, les drivers doivent nécessairement sortir par l'une des routines suivantes :

DRBSY (busy)

Le driver attend une interruption. Le contrôle retourne au programme ayant effectué la requête.

DRFIN (finished)

Cette routine est appelée par le driver au moment où toutes les interruptions ont été reçues et où l'opération est terminée. USTQ.A est remis à 0, USTQ.B et PSL.A prennent la valeur 1 si une routine d'interruption doit être exécutée.

DRSTC (stack on connect)

Un appel à XCONN ou à RCONN a été infructueux, une demande pour rappeler le driver est placée dans une pile (stack). Dès que le canal sera libéré, PRS (process stack) appellera le driver par DRIVERU.

DRSTT (stack on time)

L'unité n'est pas prête physiquement. Une demande de rentrée dans le driver est placée dans la pile. Cette rentrée aura lieu dans les 5 secondes qui suivent.

DRERR

Une erreur a été détectée par FETCW, par MOVWCW ou par le driver lui-même. L'exécutif, lorsqu'il aura le contrôle, procédera à la terminaison du travail ayant effectué la requête.

DRBAD

Un mauvais fonctionnement de l'unité a été détecté par le driver, ce qui est fatal pour le programme central, mais pas pour un programme background.

Quelle que soit la routine à laquelle le driver rend le contrôle, celui-ci est à son tour rendu à DRIVER qui a fait l'appel, le cycle est ainsi fermé.

Un appel au driver peut provenir de différents endroits. Dans tous les cas, le registre A contient le LUD, tandis que le registre Q permet de distinguer la provenance de l'appel.

Pour un appel de DRIVERU, les valeurs sont :

valeur de Q	provenance
+0	STARTUP (initialisation d'une opération);
-1	PRS (après une sortie en DRSTC);
+1	PRS (après une sortie en DRSTT).

Pour un appel venant de DRIVERL :

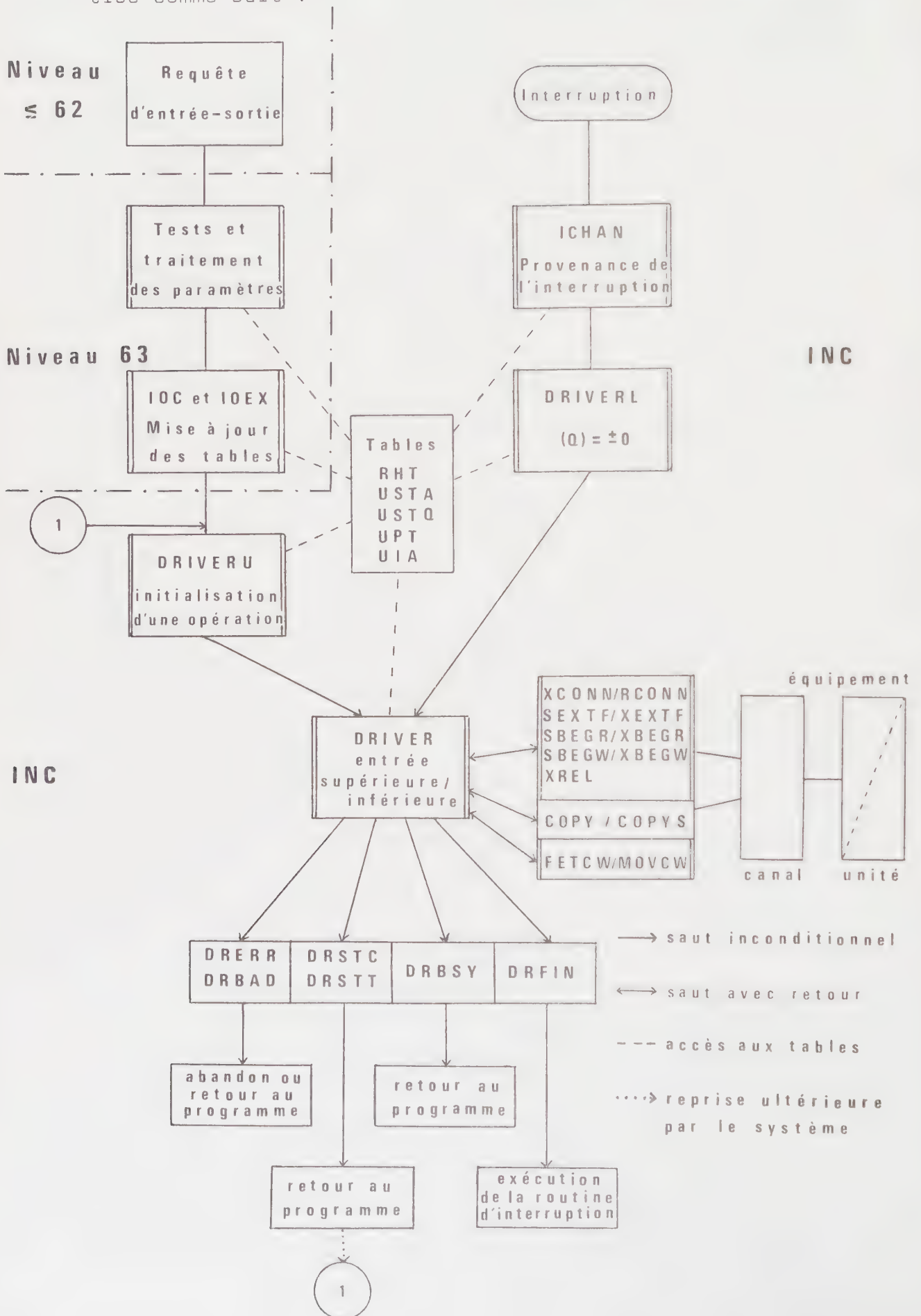
valeur de Q	provenance
+0	l'interruption vient du canal;
-0	l'interruption vient de l'équipement.

Dans ces routines d'entrée-sortie, deux points constituent des restrictions, acceptables pour un système fonctionnant en mono-programmation :

- il ne doit pas arriver d'interruption sur une unité libérée par XREL en vertu du cas 3c) de ICHAN. Cela veut dire que toute interruption doit être attendue, un appel par l'intermédiaire d'une interruption asynchrone d'un terminal n'est donc pas permis;
- il est impossible de connecter par XCONN une unité si une autre unité est déjà occupée sur le même canal et le même équipement.

Cette restriction est logique si le canal effectue un transfert, mais l'est moins si celui-ci est inactif (CHTAB.A = 0) (voir sections 10.2 et 10.3).

Le traitement d'une requête d'entrée-sortie peut être schématisé comme suit :



3.7 Traitement type d'une requête d'entrée-sortie dans le driver

Quel que soit l'équipement qu'il contrôle, le driver doit au moins effectuer les opérations suivantes :

Entrée par la demi-instruction supérieure

- Extraire de USTQ la valeur du champ USTQ.F.
- Regarder dans la table propre au driver s'il est capable de traiter cette fonction. Sinon, il s'agit d'une opération qui n'a pas de sens pour ce type d'équipement, et il suffit de sortir à DRFIN.
- La fonction a un sens, connecter l'unité par XCONN ou RCONN. Si la connection est rejetée avec (A) = -0, sortir par DRSTC; si elle est rejetée avec (A) = -1, sortir par DRBAD.
- La connection a réussi, donner le contrôle à la partie du driver traitant la fonction indiquée.
- S'il s'agit d'une opération de lecture-écriture, faire appel à MOV CW (avec (Q) = 0 pour une écriture et (Q) ≠ 0 pour une lecture). Si au retour (Q) ≠ 0, sortir par DRERR.
- S'il s'agit d'une lecture-écriture, sélectionner les interruptions sur fin d'opération normale et anormale par XEXTF, lancer l'opération par XBEGR ou XBEGW et sortir par DRBSY.
- S'il s'agit d'une opération mécanique (fonction) sur l'unité, sélectionner l'interruption sur fin d'opération anormale, initialiser l'opération mécanique par XEXTF, sélectionner l'interruption sur unité disponible et sortir par DRBSY.

Entrée par la demi-instruction inférieure

- Si (Q) = +0, prendre le mot de contrôle de fin d'opération à l'aide de l'instruction COPY et le placer dans USTA, sélectionner l'interruption sur unité disponible et redonner le contrôle à DRBSY.
- Si (Q) = -0, effacer les interruptions et mettre à jour le statut de fin d'opération. Cela se fait dans la partie du driver traitant la fonction STA. Le statut de l'équipement doit être placé dans les bits 32 à 42 de USTQ, mais aupara-

vant, les bits tombant dans la portion ayant une signification globale pour tous les équipements (bits 35 à 37, 42 et 43) doivent être déplacés s'ils ont une autre signification.

- Effectuer la sortie par DRFIN.

En cas d'erreur de parité, le driver peut essayer ou non de récupérer l'erreur. S'il ne le fait pas, ou s'il n'y parvient pas, il doit déplacer le bit de statut correspondant, s'il n'est pas déjà en position 42. Pour le reste, c'est au programme ayant effectué la requête qu'il appartient de tester USTQ et d'agir en conséquence.

Ce qui caractérise une opération de lecture-écriture type, telle que celle décrite ci-dessus, c'est qu'elle se déroule en une seule fois, qu'après son initialisation les mots de contrôle suffisent à la diriger entièrement.

Les caractéristiques des routines résidentes d'entrée-sortie imposent des règles au driver, il y a un certain nombre d'erreurs que celui-ci ne doit pas commettre. Certaines provoquent un appel à DISASTER, d'autres des boucles sans fin :

- entrer dans XCONN avec une valeur incorrecte de IORD;
- essayer de connecter par XCONN une unité occupée;
- tenter de lancer une opération sur une unité non connectée;
- libérer une unité sans que toutes les interruptions sur celle-ci aient été effacées;
- sortir à DRFIN sans avoir libéré l'unité, etc.

3.8 Autres fonctions des drivers

Dans la philosophie des drivers, deux points sont particulièrement importants :

En premier lieu, l'utilisateur doit pouvoir écrire ses instructions de lecture-écriture sans avoir à tenir compte des différences entre les types de périphériques. Pour cela, le driver assure le lien avec le système d'exploitation et le système d'interruption comme les sections précédentes l'ont montré.

Mais cela n'est pas suffisant, car d'autres différences peuvent subsister.

En effet, les équipements n'utilisent pas nécessairement le code interne de l'ordinateur et une conversion de code doit intervenir soit par hardware, soit par software. Dans ce cas, c'est le driver qui doit s'en charger.

Pour les lectures-écritures, le système DISK SCOPE connaît deux modes : binaire et BCD. Ces modes doivent avoir un sens sur chaque équipement, soit qu'ils correspondent à une réalité physique (dans ce cas, un code de fonction externe fixe le mode), soit qu'ils soient définis par le driver.

Sur les bandes magnétiques, le mode binaire correspond à la parité impaire et à la transmission dans le code interne; en mode BCD, par contre, la parité est paire et le contrôleur effectue une conversion de code.

Sur le lecteur et le perforateur, les deux modes ont également une signification hardware. Le mode BCD implique l'utilisation du code Hollerith étendu (une carte représente 80 caractères, soit 10 mots, et il n'existe que 64 perforations légales), tandis que le mode binaire correspond à l'utilisation de toutes les perforations possibles (une carte représente alors 20 mots). Pour les autres périphériques, ce sont les drivers qui donnent une signification software au mode.

Sur les disques, un indicateur de mode précède les données qui sont écrites de la même manière, quel que soit le mode.

Sur l'imprimante, le premier caractère de chaque ligne tient lieu, en mode BCD, de caractère de contrôle tandis qu'il est imprimé tel quel en mode binaire.

Sur la machine à écrire de la console, qui a un code de caractères propre, la conversion se fait si le mode est BCD et n'a pas lieu s'il est binaire.

Ainsi, l'identité entre périphériques peut être poussée assez loin pour les opérations de lecture-écriture.

Le second point important est que les caractéristiques propres de chaque périphérique doivent rester accessibles dans les autres opérations sur les périphériques. Pour cela, on peut :

- soit créer une nouvelle requête qui n'est interprétée que par un seul driver;
- soit attribuer un sens bien particulier à une requête qui n'aurait sans cela aucune signification sur l'équipement considéré.

Les requêtes ERASE (effacement d'une portion défectueuse de bande magnétique) ou LOCATE (positionnement sur un disque travaillant en accès direct) conviennent très bien pour cela (il leur correspond au niveau de IOEX et des drivers les codes ERA et LOC respectivement). Par exemple, la requête ERASE peut signifier "décaler la dernière carte perforée" sur le perforateur où cette possibilité physique existe.

Il peut également être nécessaire de définir plus de deux modes pour les lectures-écritures ou de diviser chacun des modes en "sous-modes" (c'est le cas pour le dispositif de visualisation). Une solution consiste à attribuer un sens particulier aux trois densités qui n'ont de sens que pour les bandes. Une autre solution, moins bonne, revient à effectuer les opérations standard pour les mots de contrôle IOT_R^W et spéciales pour les mots de contrôle IOS_R^W .

Grâce à ces différents artifices, les possibilités physiques de l'équipement sont accessibles au niveau du langage assembleur, il suffit d'écrire des petits sous-programmes de liaison pour les rendre accessibles aux langages supérieurs.

La plupart des solutions envisagées ci-dessus ont été utilisées dans la suite de ce travail. Leur intérêt est qu'elles ne nécessitent aucun aménagement spécial dans le système d'exploitation, qu'il suffit de fixer des conventions judicieuses et de les programmer dans le driver.

Cependant, les trois équipements présentés dans la section 2.4 ont posé des problèmes insolubles dans le cadre décrit jusqu'ici. Ces difficultés et les extensions apportées au système pour les résoudre font l'objet du chapitre suivant.

Chapitre 4 : Généralités sur les drivers des équipements du terminal

4.1 Fractionnement des opérations d'entrée-sortie

Pour un ordinateur du genre CDC 3800, les équipements peuvent être classés en trois catégories si l'on compare leurs vitesses typiques de transfert :

- les équipements rapides : bandes, disques (vitesse de transfert supérieure à 10^5 caractères/s);
- les équipements lents : lecteurs et perforateurs de cartes, imprimantes (vitesse de transfert : 300 - 3000 caractères/s);
- les équipements très lents : machine à écrire (en écriture : 15 caractères/s, en lecture : vitesse humaine).

Le dispositif de visualisation se rattache à la deuxième catégorie, la machine à écrire et le clavier à la troisième.

Les canaux sont conçus pour diriger les opérations des équipements rapides et lents, mais il est inconcevable de bloquer un canal par un équipement très lent, en attendant qu'un utilisateur ait complètement tapé un texte sur un clavier. Si l'équipement ne possède pas de mémoire tampon de plusieurs caractères, ceci implique que l'opération doit être fractionnée au niveau du caractère; par exemple, la lecture ou l'écriture d'un texte de N caractères sur la machine à écrire ne constitue pas 1, mais N appels au driver. Pour ne pas occuper inutilement le canal, il faut toutefois qu'un tel équipement ait une mémoire tampon de 1 caractère.

Ce fractionnement est une caractéristique d'un équipement interactif.

Un autre motif qui peut nécessiter le fractionnement d'une opération est le changement de code. Aucun des équipements du terminal CICE ne travaille dans le code interne de l'ordinateur (voir annexe 2). Pour un équipement très lent, l'opéra-

tion s'effectue déjà caractère par caractère, que le code soit différent du code interne ne pose aucun problème, il suffit d'introduire au moment voulu une conversion de code.

Ce point devient plus gênant pour un équipement lent, tel l'écran de visualisation : il ne travaille plus au niveau du caractère, mais va transmettre des blocs de plusieurs mots. Il faut donc adjoindre au driver une zone contenant les caractères convertis dans le code du dispositif de visualiation; il est impossible de prévoir la longueur des messages à écrire et il est gênant de les limiter par la taille de cette zone tampon. Celle-ci devra donc être remplie et vidée autant de fois que nécessaire jusqu'à la transmission complète du message, ce qui introduit donc une autre forme de fractionnement.

L'écriture de drivers pour des équipements nouveaux doit tenir compte du principe : leur utilisation doit être compatible avec celle des équipements déjà existants; seules les cartes de contrôle sont à changer.

Le fractionnement des opérations pose un premier problème au niveau des chaînes de mots de contrôle. Dans le cas type, la routine MOVCW amène une chaîne dans la zone CWORDS du canal, cette chaîne est ensuite entièrement exécutée, il n'y a donc aucune précaution à prendre. Dans le cas d'opérations fractionnées, les mots de contrôle de l'utilisateur ne peuvent pas être utilisés tels quels. Le driver travaille avec ses propres mots de contrôle en interprétant ceux de l'utilisateur, par des tests sur les bits 45-47. Cette chaîne de mots de contrôle ne peut pas être laissée dans la zone CWORDS, car elle serait détruite, lors de l'utilisation d'un autre équipement sur le même canal, avant d'avoir été complètement exécutée.

Pour conserver les mêmes caractéristiques aux opérations fractionnées ou non, chaque unité, ayant recours au fractionnement, doit disposer d'une zone de mots de contrôle.

Le second problème est le sauvetage des valeurs permettant de reprendre l'opération fractionnée là où elle a été interrompue. Certes, chaque unité logique dispose d'un mot UPT, mais

les caractéristiques hardware et les contraintes software peuvent augmenter le nombre de ces valeurs, un mot devient insuffisant. Il est exclu de garder en permanence des valeurs à l'intérieur du driver car, s'il n'est pas résident, il peut être rechargé à partir du disque, dans une portion différente de la mémoire, dès que toutes les unités contrôlées par le driver ont été libérées par XREL. Le sauvetage de ces valeurs auxiliaires ne peut se faire que dans des zones fixes, hors du driver.

4.2 Modification des tables

L'adjonction de trois périphériques provoque des modifications des tables que DISK SCOPE permet sans autre.

Il suffit d'ajouter sur la bande système les entrées supplémentaires dans HTL et AET, selon les formes symboliques externes décrites dans la section 3.4.

Les mnémoniques choisis sont : TW (typewriter), DY (display) et KB (keyboard), ce qui implique les noms des drivers : DRTWD, DRDYD, DRKBD.

Les difficultés soulevées dans la section précédente ont été résolues par l'adjonction de nouvelles tables, également créées par PREDRUM. Ici intervient un élément simplificateur : à chacun des nouveaux équipements n'est rattachée qu'une seule unité (et il n'est pas prévu d'en ajouter d'autres). Si XX désigne l'un quelconque des trois mnémoniques introduits, deux tables ont été ajoutées pour l'équipement XX.

XXCW est une zone de longueur .CWORDS + 1, contenant les mots de contrôle de l'équipement (et par conséquent de l'unité) XX. Les .CWORDS derniers mots de XXCW sont prévus pour la chaîne de mots de contrôle; après l'appel à MOV CW, le driver n'a qu'à copier le contenu de la zone du canal dans celle de l'équipement.

Le premier mot de XXCW joue un rôle triple :

- Son adresse inférieure sert de pointeur sur le mot de contrôle en cours de traitement, ce pointeur est augmenté d'une unité chaque fois qu'il y a passage au mot de contrôle suivant.
- Après l'appel à MOV CW, le driver DRXXD fait pointer ce mot sur lui-même et met les bits 45-47 à 1, ainsi le premier mot de contrôle de la chaîne de l'utilisateur est précédé d'un "mot de contrôle zéro" :

IOTR,C (0)*,0

L'avantage est que le passage au premier mot de contrôle ne requiert pas un traitement particulier, il est ramené à celui du i -ème au $(i+1)$ -ème ($i \geq 1$).

- A la fin de chaque opération, cette adresse inférieure est remise à zéro. Elle sert donc de flag, et permet de dire si l'appel à MOV CW a déjà été fait ou non, c'est-à-dire si c'est la première entrée dans le driver pour l'opération en cours.

XXAUX est une deuxième zone contenant un nombre variable de cellules auxiliaires utilisables au gré du driver.

La création de XXCW et de XXAUX s'effectue par l'intermédiaire de la forme symbolique des entrées HTL. Si le paramètre C y figure, PREDRUM crée XXCW, si le paramètre An est spécifié, PREDRUM crée la zone XXAUX de n mots ($n \leq 9$).

Ainsi, les entrées HTL des trois équipements du terminal sont :

TW,0200,I,O,C,A2

DY,0400,O,C,A7

KB,0000,I,O,C,A3

Bien entendu, PREDRUM a été modifié afin de permettre le décodage des caractères C, A et du paramètre n, ainsi que la création des zones au moment de l'allocation de la place en mémoire pour les tables.

Pour chacune de ces deux tables, le problème est simplifié par la présence d'une seule unité. Si à un équipement étaient

rattachées plusieurs unités, chacune devrait avoir sa zone de mots de contrôle et de positions auxiliaires. Dans ce cas, il y aurait une succession de blocs contigus et de longueurs respectives égales, l'adressage devrait sélectionner un bloc et à l'intérieur de celui-ci un mot. Ce problème est examiné plus en détail au chapitre 10. Dans la situation présente, la sélection du mot dans le bloc suffit.

Exemples d'adressage :

dans TWCW	LDA	TWCW,7	charger dans A le mot de contrôle en cours d'exécution
dans DY AUX :	VADDR	EQU 1	
	STA	DY AUX+VADDR	stocker le contenu de A dans le deuxième mot de la zone DY AUX

(DY AUX+VADDR est une variable auxiliaire contenant l'adresse verticale sur l'écran de visualisation; pour abréger, elle sera désignée par la suite par VADDR).

Toutes ces zones auxiliaires se trouvent dans la région des tables, elles sont donc totalement protégées.

Chapitre 5 : Le système de simulation

5.1 Mise au point des équipements et des drivers

Le développement simultané d'équipements et de leurs drivers pose passablement de problèmes, car la cause des fonctionnements anormaux n'est pas toujours facile à déceler, le premier point consiste à déterminer si la cause est dans le hardware ou dans le software. Des mises au point séparées s'imposent donc.

Le groupe hardware a construit un simulateur de canal 3806, pouvant être branché sur les équipements. Il s'adresse à eux exactement de la même manière que le ferait le canal. Le simulateur permet d'effectuer les instructions CONN, EXTF, BEGR, BEGW; les bits de statut sont affichés en permanence et un bouton poussoir joue le rôle de l'instruction CLCH. Le simulateur transmet un mot sous forme de 4 bytes de 12 bits, dont l'envoi est commandé manuellement par un bouton poussoir. La possibilité d'envoyer ces 4 bytes de manière répétitive existe également.

Dans une première phase, ce simulateur aide à vérifier le fonctionnement correct de toutes les possibilités.

Ce test ne suffit pas :

- 1) le comportement de l'équipement à pleine vitesse peut être différent de celui en mode pas à pas;
- 2) des séquences d'opérations peuvent avoir un comportement anormal, alors que les comportements individuels sont corrects.

Pour le premier point, la solution consiste à écrire sur bande magnétique des programmes en absolu à l'aide de LIBEDIT, puis de charger la bande par le processus d'autoload, ce qui permet de contrôler le fonctionnement des équipements à la vitesse usuelle, sans intervention du système.

Ces programmes doivent être aussi simples que possible, mais le traitement doit déjà se rapprocher de celui qui est prévu

pour le driver, en particulier pour tout ce qui touche au système d'interruption.

Ce procédé donne également de bons résultats pour les difficultés du deuxième genre.

Pour la partie software, les drivers font également l'objet d'une mise au point séparée. A l'intérieur du système DISK SCOPE, ils travaillent en mode d'interruption et ont accès à n'importe quelle portion de la mémoire, dans ces conditions, la détection d'erreurs est plus difficile. Pour cette mise au point, un ensemble de programmes, le système de simulation, a été conçu. Il fonctionne entièrement au niveau 0, comme n'importe quel programme d'utilisateur et fait également travailler les drivers au niveau 0. Autour des drivers, le système de simulation recrée l'environnement du niveau 63, de INC et des équipements eux-mêmes.

Ce système nécessite donc :

- un sous-programme faisant les appels au driver, simulant IOC et IOEX;
- la réécriture des routines à disposition des drivers;
- un sous-programme simulant le comportement d'un canal, du système d'interruption et de ICHAN;
- pour chaque équipement, un sous-programme simulant son comportement (et celui de l'unité);
- enfin, le programme principal, jouant le rôle de l'utilisateur et comportant la suite des opérations d'entrée-sortie simulées à exécuter.

Avant d'examiner les détails de la réalisation, qui font l'objet des sections suivantes, il est intéressant de passer en revue les avantages et les désavantages de ce système.

Les avantages sont :

- La rapidité et la facilité de la mise au point sont des éléments déterminants car les possibilités de correction d'un driver à l'intérieur de INC sont très restreintes. Pour des modifications d'une certaine importance, le seul moyen est de recompiler le driver corrigé, de recréer une

bande système à l'aide de LIBEDIT et de la charger. Cet ensemble d'opérations prend plus de 10 minutes et doit par conséquent être évité autant que possible. L'ensemble des tests en simulation prend moins de 1,5 s.

- Un programme exécuté au niveau 0 bénéficie du système d'interruption actif. Ainsi, toutes les erreurs à caractère destructif (erreurs d'indiquage ou d'adressage risquant de modifier le système d'exploitation, instructions illégales, etc.) sont trappées immédiatement, ce qui n'est pas le cas à l'intérieur de INC.

- Un programme exécuté au niveau 0 peut effectuer des impressions intermédiaires de contrôle et dispose de toutes les facilités de mise au point offertes par le système : dumps partiels en cours d'exécution ou en fin de travail.

Les erreurs courantes sont ainsi décelées assez aisément : erreurs de perforation, interversions d'instructions, destructions accidentelles du contenu des registres, erreurs de logique, etc.

En plus de l'exécution des opérations d'entrée-sortie simulées, le système de simulation effectue un contrôle du déroulement logique des opérations commandées par le driver. Le comportement incorrect vis-à-vis des routines du système est ainsi trappé.

- Lors d'une correction ou d'une modification d'un driver, il faut non seulement qu'elle ait l'effet désiré, mais également qu'elle n'altère pas ce qui fonctionnait auparavant. Un jeu complet de tests, faisant intervenir toutes les parties du driver, s'impose. S'il est facile de passer les tests d'écriture sur le système réel avec apparition des résultats sur les unités, il n'en va pas de même pour les tests de lecture : à cause de la vitesse humaine ils prennent un temps considérable; d'autre part, ils peuvent être faussés par des erreurs de frappe. La simulation permet ainsi d'éviter un ensemble de tests fastidieux sur les équipements eux-mêmes et les remplace par des résultats imprimés que l'on peut étudier en toute tranquillité.

- Lorsqu'on regarde un texte s'écrire sur le dispositif de visualisation ou sur la machine à écrire, on a sous les yeux uniquement le résultat. On ne sait pas exactement quelle succession de codes a passé à travers le canal et l'équipement (ni l'un ni l'autre ne peuvent fonctionner pas à pas). Si, par exemple, le driver envoie des séquences superflues de codes sans effet, ou dont les effets s'annulent, le résultat n'est pas altéré. Le système de simulation imprime tous les codes transmis, ce qui permet le contrôle de tous les bytes.
- Les tests sur les équipements réels ont été effectuées en grande partie pendant des heures de temps machine réservé. Dans une exploitation aussi chargée que celle de l'Université de Genève, ces heures ont été fortement restreintes à certaines périodes. Heureusement, les tests en simulation pouvaient passer à n'importe quel moment pendant l'exploitation normale.
Enfin, lorsqu'un équipement était en panne ou en réparation, le développement du software n'était pas retardé car, avec le système de simulation, tout se passe dans l'unité centrale et la mémoire de l'ordinateur.

Les désavantages du système de simulation sont :

- Ce système a dû également être écrit, testé et mis au point petit à petit. Son développement a eu lieu parallèlement à celui des drivers eux-mêmes.
- L'écriture d'un sous-programme de simulation de l'équipement allait de pair avec celui du driver (cet inconvénient n'est pas trop grave, voir section 5.4).
- Le facteur temps est totalement différent entre la réalité et la simulation. Par exemple, si le driver ne travaille pas avec les interruptions qui conviennent, la transmission simulée est correcte, tandis que dans la réalité des codes auraient pu être perdus ou recouverts avant d'avoir été transmis. Cet inconvénient est cependant mineur, car il s'agit là d'erreurs tellement manifestes dans la réalité qu'elles ont très peu de causes possibles et sont, par conséquent, faciles à localiser.

Après ces mises au point séparées du hardware et du software, le résultat idéal aurait été qu'après l'introduction du driver dans le système DISK SCOPE et le branchement des câbles sur le canal, un programme simple de lecture-écriture fonctionne correctement. En fait, cela n'a pas été le cas. Les raisons sont :

Pour le hardware :

- au moment de l'utilisation effective, des interactions entre équipements, sous forme de parasites, sont apparues.

Pour le software :

- au début de la mise au point, certaines erreurs des drivers ont passé inaperçues, ce qui a entraîné, en retour, des corrections dans le système de simulation.

Pour remédier à ces erreurs, deux possibilités ont été fréquemment utilisées :

- programme sur bande autoloading plus poussé et cernant un point particulier de plus près;
- exécutions des instructions à la console de l'ordinateur, pas à pas, et prise de dumps.

Ces deux solutions ont l'inconvénient de prendre beaucoup de temps machine.

Le système de simulation s'est avéré particulièrement utile lorsque certaines options des drivers ont été modifiées ou complétées. Plusieurs fois, ces modifications, développées en simulation, ont fonctionné correctement dès leur mise en service.

5.2 Structure générale du système de simulation

Le système de simulation, écrit en assembleur COMPASS, est construit de manière modulaire.

L'idée essentielle est d'exécuter au niveau 0 un ensemble de tests aussi complet que possible avec les drivers réels mais en remplaçant d'une part, les équipements par des sous-programmes réagissant comme ceux-ci et d'autre part, les requêtes

d'entrée-sortie d'un programme utilisateur par des appels aux sous-programmes du système de simulation décrivant les tests à effectuer.

Le système de simulation se compose des sous-programmes suivants :

- DRIVTEST est le programme principal contenant l'ensemble des tests à effectuer, ils sont écrits sous forme d'appels (requêtes) à SYSTEM.
- SYSTEM analyse les paramètres de ces requêtes et donne le contrôle aux drivers. Il simule le comportement de REQUEST, IOC et IOEX.
- Toutes les routines utilisées par les drivers sont groupées dans le sous-programme ROUTINES, il contient également la simulation du canal et du système d'interruption.
- Les sous-programmes de simulation des équipements sont TWSIMUL, DYSIMUL et KBSIMUL.
- BEGEXEC effectue les mesures de temps et les imprime.
- DUMP se charge des impressions intermédiaires, essentiellement des statuts.
- Enfin COMPARE contrôle en fin de travail les résultats de la simulation.

Le programme DRIVTEST

Les requêtes au système de simulation sont très semblables à celles au système DISK SCOPE. La différence essentielle réside dans le fait que la requête à DISK SCOPE fait intervenir une zone de mémoire et une unité tandis que la requête à SYSTEM fait intervenir deux zones de mémoire : il y a bien sûr la zone vers laquelle pointe le mot de contrôle et une nouvelle zone, appelée zone auxiliaire, où sont stockés les bytes qui arrivent à l'unité ou qui en viennent.

Dans le cas de la lecture simulée, le transfert se fait de la zone auxiliaire vers celle désignée par le mot de contrôle, dans le cas de l'écriture simulée, le transfert se fait dans le sens inverse.

Par analogie avec les requêtes à DISK SCOPE, celles à SYSTEM se composent d'une instruction de branchement contenant un code identifiant l'appel et d'un certain nombre de paramètres.

Ces requêtes sont codées sous forme de macros et un paramètre d'assemblage permet de générer des appels soit à SYSTEM, soit à DISK SCOPE.

Les paramètres d'une requête à SYSTEM sont :

u : unité (chaque unité est désignée par son numéro physique, les numéros logiques et physiques sont confondus);
cwa : adresse du mot de contrôle;
buf : adresse de la zone auxiliaire.

Les requêtes à SYSTEM sont :

SREAD	u,cwa,buf	lecture
SWRITE	u,cwa,buf	écriture
SMARKEF	u,buf	écriture d'une fin de fichier
SRELEASE	u,buf	libération de l'unité
SDYSTAT	u	statut dynamique
SSTATUS	u	statut
SMODE	u,p ₁ ,p ₂ ,p ₃ ,p ₄	définition des modes de fonctionnement
SETUBUF	u,buf	modification de l'adresse de la zone auxiliaire
SEXIT		libération de toutes les unités
SETBIT	u,b,d	
CLEARBIT	u,b,d	

Ces deux dernières requêtes permettent de simuler des conditions anormales ou asynchrones pendant le déroulement d'une opération simulée. Ces requêtes précèdent celles d'entrée-sortie, leur effet intervient pendant l'exécution de ces dernières. SETBIT et CLEARBIT permettent respectivement de mettre ou d'effacer l'un des 12 bits liés au statut de l'équipement.

u : unité;
b : numéro du bit de statut;
d : délai (en ms) avant la modification du statut.

Parallèlement à chaque zone dans laquelle un driver stocke ses résultats, une autre zone contenant les résultats théoriques a été ajoutée. Celle-ci n'intervient pas lors des requêtes mais, en fin de travail, la routine COMPARE vérifie les résultats

obtenus et théoriques mot par mot et imprime un message en cas de non-concordance. L'intérêt est que lors d'une modification d'un driver, il est possible de voir très rapidement si elle a eu les effets escomptés et si, d'autre part, elle n'a pas introduit des effets inattendus.

Le sous-programme SYSTEM

SYSTEM reçoit les requêtes, extrait et analyse leurs paramètres, met à jour les cellules et les tables, effectue l'assignation au besoin, établit la communication avec le driver et le programme de simulation concernés et donne le contrôle au driver.

C'est dans SYSTEM que sont stockés les masques utilisés dans les opérations logiques, ainsi que les tables. Chaque unité dispose d'un mot dans chacune des tables suivantes : RHT, USTA, USTQ, UPT, UBUF (adresse de la zone auxiliaire), STAT (statut physique et logique).

Dans le système de simulation, tout se passe sur un seul et même niveau, ainsi un seul jeu de pointeurs est nécessaire : IRHT, IUSTA, IUSTQ, IUPT, IUBUF, ISTAT contiennent les adresses des mots dans les tables, IORD et ILUD sont également définis. Un paramètre d'assemblage permet d'introduire une zone de .CWORDS + 1 mots de contrôle XXCW et une zone de n cellules XXAUX.

Comme sous DISK SCOPE, une requête peut provoquer plusieurs appels au driver, en particulier si une assignation a lieu. L'assignation simulée est beaucoup plus simple que celle sous DISK SCOPE : elle ne comporte qu'un appel au driver avec la fonction SDH (que celui-ci peut interpréter pour effectuer des initialisations) et la mise à jour du champ RHT.K.

L'appel au driver est également simplifié : puisque tout se passe sur un seul niveau, il n'y a pas lieu de simuler la requête STARTUP. SYSTEM place le code de l'opération dans le champ USTQ.F et donne le contrôle à la demi-instruction supérieure du premier mot du driver concerné.

Les drivers

Les drivers sont les seuls sous-programmes identiques sous le système de simulation et sous le système réel. Il faut remarquer que les drivers n'utilisent jamais les instructions d'entrée-sortie (qui sont interdites au niveau 0) mais toujours des branchements à des routines (voir section 3.6) qui sont simulées (section 5.3).

Le fait d'avoir au niveau 0 des instructions interdites ne constituerait de toute façon pas un obstacle : DISK SCOPE permet de demander, lors d'une interruption sur instruction illégale, le débranchement dans une routine déterminée. Celle-ci peut alors rechercher l'instruction illégale, la simuler ou l'interpréter, puis rendre le contrôle à l'instruction suivant l'instruction illégale.

5.3 Le sous-programme ROUTINES

Ce sous-programme

- contient toutes les routines à disposition des drivers;
- effectue la simulation des états logiques et contrôle ces états;
- simule le fonctionnement d'un canal;
- contient quelques routines à disposition des sous-programmes de simulation des équipements, en particulier la simulation des interruptions.

Les routines à disposition des drivers, simulées dans ROUTINES, sont :

XCONN, RCONN, RECONN, XEXTF, SEXTF, XBEGR, SBEGR, XBEGW, SBEGW, COPY, COPYS, XREL, FETCW, MOVW, DRSTC, DRSTT, DRFIN, DRERR, DRBAD.

L'entrée STAT de chaque unité contient deux statuts : le statut physique de 12 bits (+ 1 bit pour marquer les interruptions du canal), contenu dans la portion adresse inférieure, et le statut logique contenu dans la portion adresse supérieure.

Ce statut peut prendre les valeurs :

- 0 unité libérée
- 1 unité logiquement connectée
- 2 unité logiquement occupée
- 3 unité logiquement occupée avec canal actif.

Ces valeurs sont constamment mises à jour dans les routines d'entrée-sortie simulées. La connection physique est simulée en établissant la communication avec le sous-programme de simulation de l'équipement et la déconnection en la remplaçant par un branchement dans la routine d'erreur.

Les différentes routines détectent les fautes suivantes que peuvent commettre les drivers et appellent la routine d'erreur.

- la valeur de IORD est hors des limites;
- la routine XCONN est appelée pour une unité occupée;
- une opération est tentée sur une unité non connectée;
- une interruption se produit sur une unité libérée;
- une unité est libérée, sans que toutes ses interruptions aient été effacées;
- une unité n'a pas été libérée à la fin de l'opération;
- le driver envoie un mot de contrôle dont le compte de mots est nul.

La routine FETCW effectue pour le compte des drivers un certain nombre de tests sur les mots de contrôle en provenance de DRIVTEST. Ces tests peuvent donner lieu aux messages d'erreur suivants :

- le mot de contrôle détruit la zone de mémoire protégée;
- le mot de contrôle contient une adresse inexistante;
- la chaîne de mots de contrôle est trop longue.

Lorsqu'un driver appelle les routines XBEGR, XBEGW, SBEGR ou SBEGW, il donne l'adresse d'une chaîne de mots de contrôle. Le sous-programme simulant le fonctionnement du canal prend ces mots de contrôle l'un après l'autre en cas de chaînage et les transmet au sous-programme de simulation de l'équipement. Les commandes IOJP à l'intérieur de la chaîne sont également exécutées par ce sous-programme.

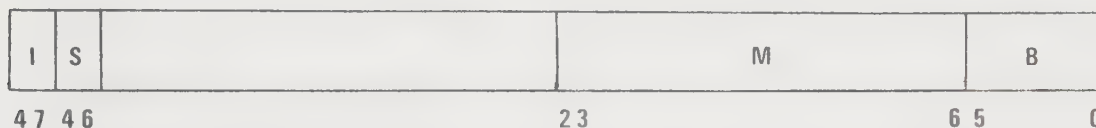
Enfin, pour compléter les tests des drivers, deux éléments

aléatoires ont été introduits : le succès de la connection et la déconnection possible d'un équipement après la fin d'activité et l'interruption du canal.

En plus des sous-programmes à disposition des drivers, ROUTINES contient des routines destinées à simplifier l'écriture des sous-programmes de simulation des équipements :

DISCONN établit la déconnection physique, c'est-à-dire remplace le branchement dans le sous-programme de simulation de l'équipement par un branchement dans la routine d'erreur.

Les sous-programmes simulant les interruptions utilisent une table de 5 mots : à chaque mot correspond une interruption possible.



Le bit 47 marque la présence de la condition d'interruption, le bit 46 la sélection. Les bits 6 à 23 contiennent un mnémorique de 3 caractères identifiant l'interruption et les bits 0 à 5 le numéro du bit de statut associé. Le bit de statut 14 correspond à l'interruption de fin de chaîne, provenant du canal, cette convention est commode pour la simulation, mais ne correspond pas à la réalité.

Les routines gérant cette table sont :

INTSLCT création d'une entrée dans la table et mise à 1 du bit 46.

INTER mise à 1 du bit 47 et, si le bit 46 vaut également 1, mise à 1 du bit de statut associé puis appel au driver par DRIVERL.

INTREL remise à 0 des bits 46, 47 et du bit de statut associé.

Ces trois routines sont appelées par les sous-programmes de simulation des équipements. Dans les trois cas, les paramètres sont le mnémorique de 3 caractères et le numéro du bit de statut associé. Il faut noter que l'arrivée de ces interruptions ne présente aucun élément aléatoire dans le temps mais suit une séquence bien définie dans chaque sous-programme de simulation

d'équipement.

Dans cette partie liée au traitement des interruptions, trois conditions d'erreur, pouvant provenir des sous-programmes de simulation d'équipement, sont testées :

- la longueur de la table d'interruption est dépassée;
- le bit de statut associé à une interruption est incorrect;
- le sous-programme de simulation revient dans ROUTINES au lieu d'appeler le driver.

5.4 Sous-programmes de simulation des équipements

Les sous-programmes décrits jusqu'ici sont soumis à des contraintes bien précises puisque leur but est de simuler un système d'exploitation existant. En revanche, les contraintes des sous-programmes de simulation des équipements ne sont pas définies de manière aussi stricte. Leur rôle est de fournir suffisamment de renseignements pour permettre de vérifier aussi aisément que possible si tout fonctionne correctement; puisque ce sont eux qui transfèrent et stockent les données selon les indications des drivers.

Un sous-programme de simulation d'équipement n'est pas tenu de transférer exclusivement ce qui est commandé par le driver, mais il peut ajouter des informations complémentaires permettant de contrôler d'autres éléments relatifs à l'état du périphérique par exemple (c'est le cas de DYSIMUL).

Ces programmes doivent être aussi simples que possible, c'est pourquoi les routines communes à tous ont été regroupées dans ROUTINES, ainsi beaucoup d'opérations se ramènent à de simples séquences d'appel.

D'autre part, ces sous-programmes reflètent non pas le fonctionnement aussi général que possible des équipements, mais sont partiellement écrits en fonction de la manière dont ils vont être appelés par les drivers.

Ces sous-programmes sont appelés par ROUTINES exclusivement, à chaque appel est lié un code indiquant la provenance et l'opération à effectuer.

Ces codes sont :

.EXTF	appel de SEXTF
.BEGR	appel de SBEGR
.BEGW	appel de SBEGW
.BSY	appel de DRBSY
.FIN	appel de DRFIN

Chaque code est traité dans une partie séparée à l'intérieur du sous-programme de simulation.

La partie liée à .EXTF se ramène uniquement à des appels à INTSLCT, INTREL et DISCONN (routine de déconnection physique), selon la valeur du code de fonction en provenance du driver. Les seules parties présentant quelques difficultés de programmation sont celles traitant .BEGR et .BEGW et effectuant les transferts de données. Ce sont les seules routines ayant accès aux zones vers lesquelles pointe IUBUF. Ces routines travaillent byte par byte et font au fur et à mesure la mise à jour des bits de statut physique.

Pour les opérations d'écriture, les données sont transmises vers la zone auxiliaire, le sous-programme de simulation est chargé, en outre, de disposer les résultats à l'intérieur de la zone auxiliaire afin de faciliter la lecture d'un éventuel dump.

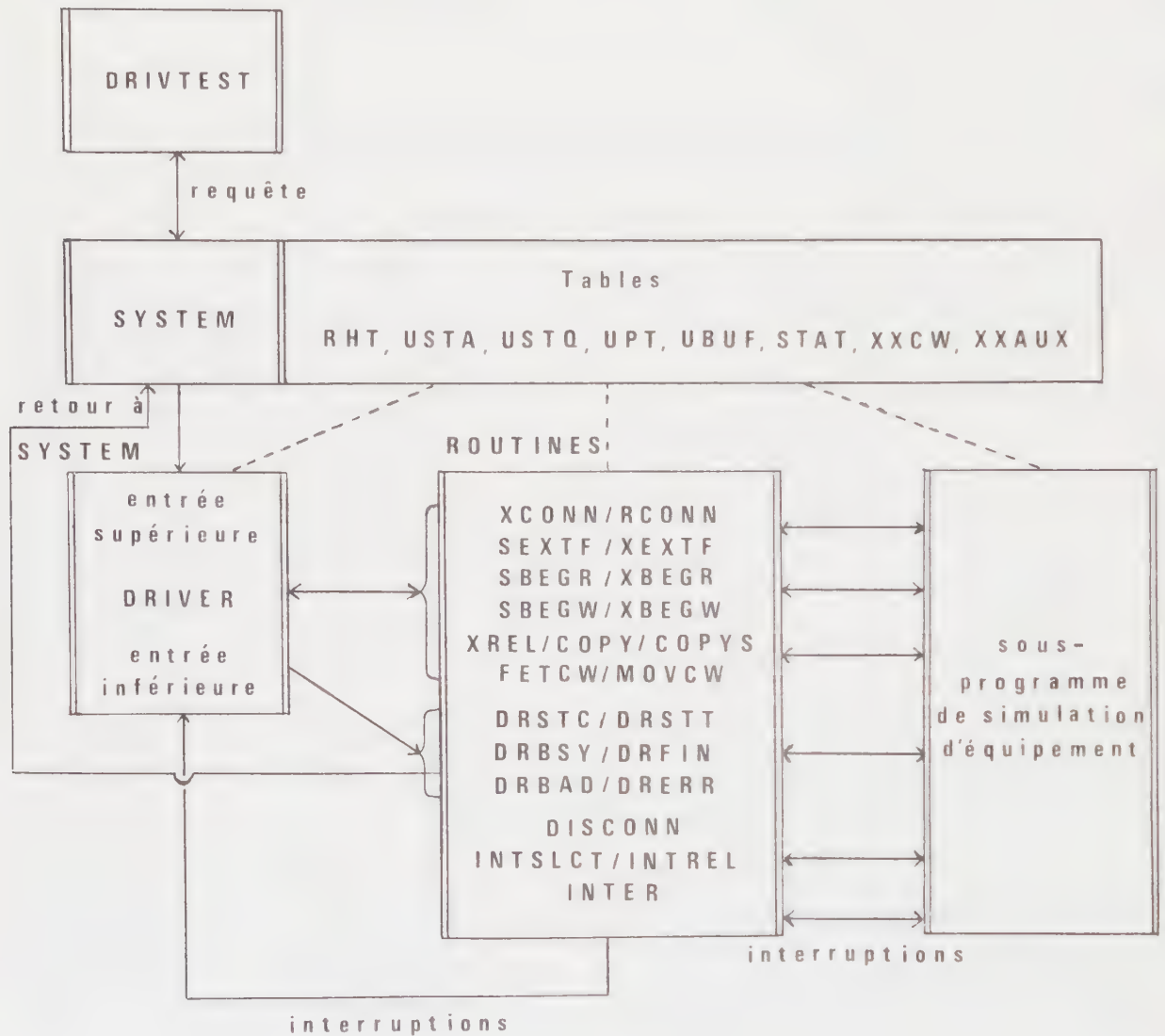
Une différence importante entre la réalité et la simulation apparaît dans la séquence des opérations :

Réalité	Simulation
- sélection d'interruptions;	- sélection d'interruptions;
- appel à XBEGR ou XBEGW;	- appel à XBEGR et XBEGW et
- sortie à DRBSY;	transfert complet;
- fin du transfert physique	- sortie à DRBSY;
et interruptions;	- simulation des interruptions
- entrées dans le driver	et entrées dans le driver
(demi-instruction infé-	(demi-instruction infé-
rieure).	rieure).

DRBSY, dans ROUTINES, appelle donc le sous-programme de simulation avec le code .BSY. Celui-là simule une interruption par un appel à INTER, qui à son tour appelle le driver par sa demi-instruction inférieure.

Dans la partie du sous-programme traitant le code .FIN, aucune activité physique n'est simulée, seules sont effectuées des remises à zéro avant l'opération suivante. C'est en particulier le cas pour la table contrôlant les interruptions.

Schéma de fonctionnement du système de simulation



→ saut inconditionnel

↔ saut avec retour

---- accès aux tables

Chapitre 6 : La machine à écrire

6.1 Caractéristiques de l'équipement

La machine à écrire est du type IBM Selectric 731, elle travaille exclusivement dans le code Selectric (il s'agit d'un code à 6 bits, voir annexe 2).

La vitesse d'écriture est d'environ 15 caractères/s. Les caractères peuvent être tapés en majuscule ou en minuscule, les chiffres et quelques caractères spéciaux sont également disponibles. Les codes spéciaux, correspondant à des actions mécaniques seulement, sont :

- espace;
- recul (espace en arrière);
- tabulateur;
- retour de chariot et passage à la ligne suivante;
- mise du clavier en mode majuscule ou en mode minuscule;
- sélection du ruban noir ou rouge.

Les codes de fonctions pour l'instruction EXTf sont (en octal) :

0	déconnection	
1	sélection	} de l'interruption sur fin d'opération anormale
2	libération	
5	libération	de toutes les interruptions
20	sélection	} de l'interruption sur unité disponible (ready and not busy)
21	libération	
22	sélection	} de l'interruption sur fin d'opération
23	libération	
100	sélection	} de l'interruption sur caractère frappé
200	libération	

La fin d'opération anormale signifie qu'un code illégal a été envoyé au contrôleur.

La condition unité disponible correspond à la fin de l'activité de la partie mécanique.

La fin d'opération et l'interruption du canal sur fin de chaîne sont pratiquement simultanées.

L'interruption sur caractère frappé n'a de sens qu'en lecture. Le code du caractère frappé, stocké dans la mémoire tampon de 1 caractère, peut alors être transmis à la mémoire centrale par l'instruction BEGR, avec un mot de contrôle dont le compte de mots vaut 1. Le caractère est transmis dans les bits 0 à 5, le reste du mot est complété par des zéros.

L'instruction BEGR peut être utilisée sans être précédée de la fonction 100₈. Dans ce cas le canal reste bloqué jusqu'à épuisement du mot de contrôle. Lorsque la fonction 100₈ ou l'instruction BEGR a été envoyée, une lumière de contrôle "type in" s'allume.

Si, lors d'une lecture, un caractère est frappé avant que le précédent ait été lu, ou si deux caractères sont frappés simultanément, un signal acoustique se fait entendre, un bouton permet de l'interrompre et d'effacer la condition d'erreur, le caractère fautif est ignoré.

En écriture, il y a également une mémoire tampon de 1 caractère. L'envoi d'un mot contenant 8 caractères bloque le canal jusqu'à ce que le 8-ème code ait atteint la mémoire tampon, soit presque une demi-seconde.

Le fractionnement consiste à envoyer un seul caractère dans les bits 0 à 5 d'un mot, complété par des zéros. Ceux-ci sont ignorés et le code du caractère est transmis à la mémoire tampon en quelques μ s, ce qui libère le canal. Le caractère suivant n'est alors envoyé qu'après réception de l'interruption sur unité disponible.

Les bits de statut sont :

- 0 prêt
- 1 occupé
- 2 clavier en mode majuscule
- 3 interruption sur caractère prêt

- 5 fin de ligne
- 6 statut manuel (un bouton spécial permet de l'enclencher)
- 7 interruption sur unité disponible
- 8 interruption sur fin d'opération
- 9 ruban rouge
- 10 interruption sur fin d'opération anormale

Aucun bit de statut de l'équipement n'est lié à l'erreur de parité. Celle-ci est testée dans la partie du terminal commune à tous les équipements. Il s'agit donc d'un test de la transmission entre le canal et l'équipement, une éventuelle erreur est signalée au canal, elle provoque la mise à 1 du bit de statut 14 et elle est détectée ultérieurement lors de l'exécution de l'instruction CPJ (sections 2.3 et 3.6).

6.2 Le driver DRTWD

Le driver DRTWD traite les fonctions RDF, WRF, STA, REL, WEF, RDY (voir section 3.5). Ces fonctions sont liées respectivement aux requêtes READ, WRITE, DYSTAT, RELEASE, MARKEF et READY.

STA et RDY correspondent à une simple consultation de statut, toutes les sorties du driver se font à travers STA.

WEF et REL sont interprétés comme 4 sauts de ligne. Le système envoie automatiquement ces fonctions en fin de travail, l'utilisateur peut ainsi se rendre compte d'une fin de travail anormale, même si aucun message n'est tapé.

Pour la lecture et l'écriture, deux modes de fonctionnement sont définis : le mode binaire et le mode BCD. Ils ont une signification software uniquement : le mode binaire correspond à l'utilisation du code Selectric, le mode BCD à celle du code BCD interne, commun à tous les autres périphériques.

Pour ce mode, une table de conversion a été construite, elle se compose de 11 mots de 6 bytes de 8 bits. Le $(k + 1)$ -ème byte contient dans ses bits 0 à 5 le code Selectric du caractère dont le code BCD est k. Les deux autres bits indiquent la position du clavier :

- 00 le caractère peut être tapé en majuscule ou en minuscule;
- 01 le caractère doit être tapé en majuscule;
- 10 le caractère doit être tapé en minuscule.

Ecriture

Le driver fournit quelques facilités quel que soit le mode :

- Lorsque le chariot arrive en bout de ligne (marqué par une marge réglable), un retour de chariot est inséré.
- Si l'utilisateur appuie sur le bouton de statut manuel lorsqu'une écriture est en cours, elle est aussitôt terminée; cela permet un gain de temps si la machine tape un grand nombre de lignes inutiles, à la suite d'une erreur. L'effet de ce bouton dure aussi longtemps qu'il est appuyé, toutefois les sauts de ligne envoyés par WEF ou REL sont toujours effectués.
- Les espaces font l'objet d'un traitement particulier : si un texte est terminé par N espaces, N déplacements du chariot n'apportent rien et constituent une perte de temps. Au fur et à mesure qu'ils arrivent, les espaces sont comptés, ce compte est stocké dans une position auxiliaire, BLCOUNT. Ils ne sont effectués que lorsqu'un autre caractère se présente. Un passage à la ligne remet BLCOUNT à 0. Au cas où l'utilisateur désire faire apparaître des espaces en fin de ligne, il lui suffit de les faire suivre, en mode binaire du code 00 (code sans effet) et en mode BCD du code 36, (converti par la table en 00).

Pour le reste, en mode binaire, les caractères donnés en code Selectric sont transmis tels quels à l'équipement. L'utilisateur doit programmer lui-même la position du clavier, il a accès à tous les codes. La pseudo-instruction de l'assembleur COMPASS, TYPE, sert à définir des chaînes de caractères en code Selectric.

Le mode BCD, le plus généralement utilisé, présente des différences importantes :

- Il y a une conversion du code interne en code Selectric, l'utilisateur n'a pas accès à quelques caractères spéciaux.
- Il n'a pas non plus à se préoccuper du mode du clavier, sans précision de sa part, les lettres sont tapées en majuscule, toutefois, l'insertion de deux caractères spéciaux lui permet de contrôler le mode d'impression des lettres : 55₈ pour les majuscules et 56₈ pour les minuscules. Chacun de ces deux codes reste en action aussi longtemps qu'il n'a pas été explicitement annulé ou qu'une nouvelle opération n'a pas commencé.
- Seul le ruban rouge est utilisé.
- Le premier caractère sert généralement de caractère de contrôle, agissant avant l'impression (pre-print spacing) : les caractères 0 et 1 provoquent respectivement 2 et 4 sauts de ligne avant l'impression, le caractère - n'en provoque aucun (le chariot continue là où il s'est arrêté), tout autre caractère produit un simple passage à la ligne.

Sur l'imprimante, le caractère 1 provoque un saut de page (une telle convention n'aurait pas beaucoup de sens sur une machine à écrire), le caractère 0 a le même sens que sur la machine à écrire, le caractère - n'a pas de sens puisque l'imprimante travaille au niveau de la ligne et non pas du caractère.

Interprétation des mots de contrôle en écriture

En mode BCD, dans une chaîne, le premier caractère ne sert de caractère de contrôle que si le mot de contrôle précédent avait le bit USTA.R (46) à 1. Cette convention est la même que pour l'imprimante.

Grâce au "mot de contrôle zéro"

IOTR,C (0)*,0

placé devant la chaîne (voir section 4.2) le premier caractère dans une opération est toujours traité comme un caractère de contrôle. A la fin d'une chaîne, les mots de contrôle IOTR et IOTW sont équivalents.

En mode binaire, IOTR et IOTW sont toujours équivalents.

Les mots de contrôle IOS_W^R ne sont pas ignorés : ils provoquent l'impression de zéros dont le premier sert, en mode BCD, de caractère de contrôle.

Cette convention peut sembler arbitraire, mais se justifie pour trois raisons :

- le hardware procède de même au cas où un mot de contrôle IOS_W^R est donné au canal;
- cette convention a été également adoptée par le constructeur pour l'imprimante;
- si ces mots de contrôle étaient ignorés, il y aurait là un moyen de tourner la convention relative au caractère de contrôle.

Lecture

En lecture également, certaines conventions s'appliquent aux deux modes :

- Le ruban noir est sélectionné pour toute lecture.
- Les caractères, au fur et à mesure de leur arrivée, sont placés directement dans la zone indiquée par l'utilisateur. Avant le début de l'opération, cette zone est entièrement remplie d'espaces par le driver.
- Deux caractères de correction sont à disposition :
 - Ø annule tout ce qui a été transmis par le mot de contrôle en cours et recommence son exécution;
 - @ annule le dernier caractère transmis et le remplace par un espace.

En mode binaire, tous les caractères sont transmis dans le code Selectric, y compris les changements de position du clavier, l'utilisateur doit procéder lui-même au décodage.

En mode BCD, tous les codes passent à travers la routine de conversion, celle-ci tient compte de la position du clavier, ses changements de position ne sont pas transmis.

Interprétation des mots de contrôle en lecture

Pour autant que $USTA.R = 1$, le retour à la ligne et le recul sont interprétés comme des marques de fin de message. Un re-

tour à la ligne, précédant le premier caractère, est considéré comme un élément de mise en page et ne provoque pas la fin de l'opération.

Un mot de contrôle IOS_W^R demande à l'utilisateur de taper un texte, celui-ci n'est toutefois pas transmis. Pour les mots de contrôle IOS_S^T , seul l'épuisement du compte de mots amène la fin de la lecture. De même pour un mot IOS_S^T , elle se termine si aucun caractère de fin de message n'a été donné, au moment où le compte de mots est ramené à 0.

Mot de contrôle de fin d'opération

Le mot de contrôle de fin d'opération n'est pas celui donné par le canal, mais il est calculé à partir de celui de l'utilisateur. Il se présente toujours comme s'il avait été complètement exécuté, c'est-à-dire le compte de mots est ramené à 0, et l'adresse des données ajustée en conséquence. Cette convention est appliquée même si, en écriture, l'impression est interrompue, ou si, en lecture, une marque de fin de message survient avant l'épuisement du mot de contrôle. La raison d'être de cette convention est qu'elle permet d'éviter des incompatibilités avec les routines d'entrée-sortie du FORTRAN.

Variables utilisées par DRTWD

Dans une opération fractionnée en caractères, la notion de compte de mots n'est pas très pratique, DRTWD la remplace par celle du compte de caractères (= 8 * compte de mots) et place cette valeur dans UPT. En lecture comme en écriture, c'est UPT qui est décrémenté et qui permet de voir si l'opération est terminée ou non.

DRTWD utilise une zone TWCW pour les mots de contrôle et une zone TWAUX de 2 mots.

Le premier mot contient, lors d'une écriture, le compte des espaces (BLCOUNT) et l'option (majuscule ou minuscule) de l'utilisateur pour l'impression des lettres (CASE). Ce même mot contient, lors d'une lecture, un flag (FSTCHAR) indiquant si un premier caractère a déjà été transmis à l'utilisateur,

ce qui permet de décider si un retour à la ligne marque une fin de message ou non.

Le deuxième mot (INDX) contient les pointeurs vers la zone de l'utilisateur : l'adresse d'un mot et la position du caractère à l'intérieur de celui-ci.

Utilisation du système d'interruption

Tous les transferts sont dirigés par le mot de contrôle
IOTR (0)CHAR,1

En écriture, DRTWD sélectionne l'interruption sur unité disponible, appelle XBEGW et sort par DRBSY. 60 ms séparent les interruptions sur fin de chaîne et sur unité disponible. Lorsque cette deuxième interruption arrive, elle est effacée et le transfert d'un nouveau caractère est initialisé. Pour une configuration plus compliquée (voir section 10.3), il faudrait introduire, avant d'initialiser le transfert d'un nouveau caractère, un saut à DRSTC afin d'intercaler une éventuelle opération en attente.

Dans la version actuelle, l'écriture d'un caractère demande deux passages à travers les routines d'interruption du système.

En lecture, DRTWD sélectionne l'interruption sur caractère frappé et sort par DRBSY. A la réception de l'interruption, un appel à SBEGR provoque la lecture effective; une boucle d'attente, testant le bit de statut 12, permet de déterminer la fin de l'opération. Cette lecture ne dure que quelques μ s et il ne vaut pas la peine de recourir au système d'interruption dans ce cas. Un paramètre d'assemblage permet toutefois d'utiliser l'interruption sur fin d'opération, mais le traitement prend beaucoup plus de temps.

Après l'effacement de l'interruption sur caractère prêt, un nouveau caractère est immédiatement demandé (pour autant qu'il n'y ait pas de condition de fin d'opération).

Dans la version standard, la lecture d'un caractère ne coûte donc qu'un seul passage à travers les routines d'interruption.

Chapitre 7 : Le dispositif de visualisation

7.1 Caractéristiques de l'équipement

Le dispositif de visualisation se compose de trois parties : le dispositif de commande, la mémoire temporaire et le dispositif de lecture [1].

Le dispositif de commande comprend 2 registres d'adresse (horizontale et verticale) de 12 bits et 10 autres registres de 8 bits : 4 registres de caractère, 2 d'incrément d'adresse, 2 de marge, 2 de commande, 1 registre auxiliaire et 1 de remise à zéro.

Dans la version actuelle, les trois registres de caractères spéciaux ne sont pas utilisés, au registre de "caractères alphanumériques simples" est associé un générateur de caractères, travaillant en code ASCII à 6 bits (64 possibilités). Le dispositif de commande traite également les fonctions et les bits de statut.

La mémoire temporaire est constituée d'un tube électronique à mémoire électrostatique, ayant trois régimes de fonctionnement s'excluant mutuellement : l'écriture, l'effacement et la lecture. Tous trois établissent la haute tension nécessaire à son fonctionnement.

Le dispositif de lecture est constitué par deux moniteurs de télévision affichant le contenu de la mémoire.

Cette formule confère à l'ensemble une très grande souplesse d'utilisation, des caractères et des figures peuvent être affichés, l'effacement dispose de toutes les possibilités de l'écriture.

Codes de fonctions pour l'instruction EXTF (en octal)

- | | |
|---|---|
| 0 | déconnection |
| 5 | libération de toutes les interruptions et remise à zéro de tous les registres |

20	sélection	}	de l'interruption sur unité disponible
21	libération		
22	sélection	}	de l'interruption sur fin d'opération
23	libération		
100	sélection	}	de l'interruption manuelle
200	libération		

Bits de statut

0	prêt
1	occupé
2	haute tension établie
5	interruption manuelle
6	fin de ligne
7	interruption sur unité disponible
8	interruption sur fin d'opération
9	fin de page

Le dispositif de commande reçoit des bytes de 12 bits, il comporte une mémoire tampon de 1 byte. Les bits 8 à 11 déterminent la nature de la commande (ou contrôle), les 8 autres constituent son argument. Les contrôles sont numérotés de 0 à 15. Ces contrôles peuvent être considérés comme le langage de programmation du dispositif de visualisation.

Contrôle 13



Ce contrôle établit les modes de fonctionnement essentiels du dispositif.

bits 0-1	: mode 00	haute tension coupée (repos)
	01	affichage (phase passive, l'utilisateur prend connaissance du contenu de l'écran)
	02	effacement
	03	écriture
		} (phases actives, le contenu de la mémoire est modifié)
bit 2	: en mode effacement,	provoque l'effacement rapide

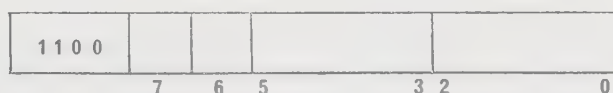
- bit 5 : mode vectoriel (ce mode permet le tracé de figures)
bit 6 : activation des registres d'adresse (pour que les modifications d'adresse soient effectives, il faut que les registres soient actifs)
bit 7 : balayage de la mémoire électrostatique.

Exemples :

- 6401 mise en lecture
6503 mise en écriture
6606 effacement de tout l'écran

La notion d'activation des registres d'adresse correspond à une contrainte du hardware; vue sous l'angle de la logique, elle peut pratiquement être ignorée.

Contrôle 12



Ce contrôle complète le contrôle 13 pour les opérations d'écriture et d'effacement.

- bits 0-2 : intensité (comprise entre 0 et 7)
bits 3-5 : taille des caractères (comprise entre 0 et 7)
bit 6 : retour automatique à la ligne
bit 7 : retour automatique en haut de page.

Contrôles 8 - 10 : adresses absolues

Le contenu des bits 0 à 7 est transféré dans les bits 4 à 11 du registre d'adresse verticale (contrôle 9), d'adresse horizontale (contrôle 10) ou des deux registres (contrôle 8).
En mode vectoriel, un point est allumé à l'adresse ainsi définie.

Contrôles 4 - 6 : adresses relatives

Le bit 7 est le bit de signe, il vaut 1 pour un incrément positif et 0 pour un incrément négatif. Les 6 autres bits constituent la valeur absolue de l'incrément.

En mode non vectoriel, le contenu des bits 0 à 6 est ajouté algébriquement aux bits 4 à 11 du registre d'adresse verticale (contrôle 5), d'adresse horizontale (contrôle 6) ou des deux registres (contrôle 4). En mode vectoriel, le contenu des bits 0 à 6 est ajouté algébriquement aux bits 0 à 6 des mêmes registres, un point est allumé à l'adresse ainsi définie.

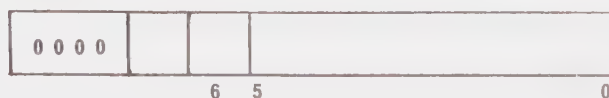
Contrôles 7 et 11 : marges

Les marges contrôlent les conditions "fin de ligne" et "fin de page". Les 8 bits les plus significatifs des registres d'adresse sont comparés aux 8 bits des registres de marge. Si le contenu d'un registre d'adresse est supérieur ou égal à la marge correspondante, une condition de fin de ligne (ou de fin de page) a lieu. Si les registres d'adresse ne sont pas actifs, ils se comportent, lors du test, comme s'ils contenaient la valeur 0.

Lors d'une remise à zéro, la marge horizontale est fixée à 360_8 ($=240_{10}$) et la marge verticale à 212_8 ($=138_{10}$).

L'argument du contrôle 7 est transféré dans le registre de marge verticale, celui du contrôle 11 dans le registre de marge horizontale.

Contrôle 0 : caractères alphanumériques simples



bits 0-5 : code ASCII de l'un des 64 caractères (voir annexe 2).
Il s'agit en fait d'un code ASCII tronqué (pour des raisons de compatibilité avec les autres équipements, un code à 7 bits n'aurait pas présenté un grand intérêt).

bit 6 : ce bit vaut 1 seulement pour certaines fonctions de mise en page qui sont :

- 160₈ : espace
- 161₈ : recul
- 163₈ : retour au début de la ligne suivante
- 164₈ : remise à zéro du registre d'adresse verticale
- 165₈ : espacement vertical

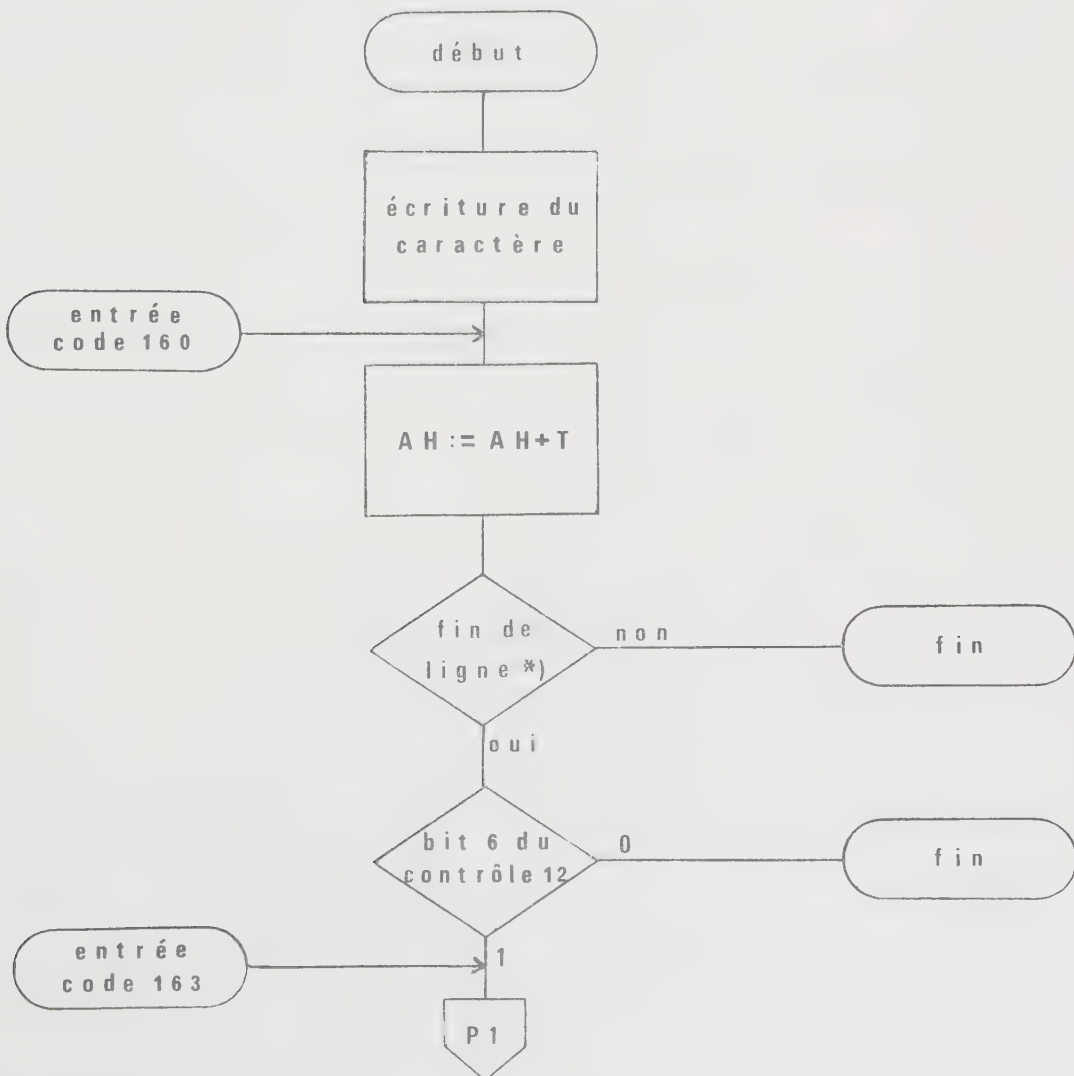
Avant l'écriture d'une suite de caractères, les contrôles 12 et 13 doivent être fixés, en particulier, les registres d'adresse doivent être actifs.

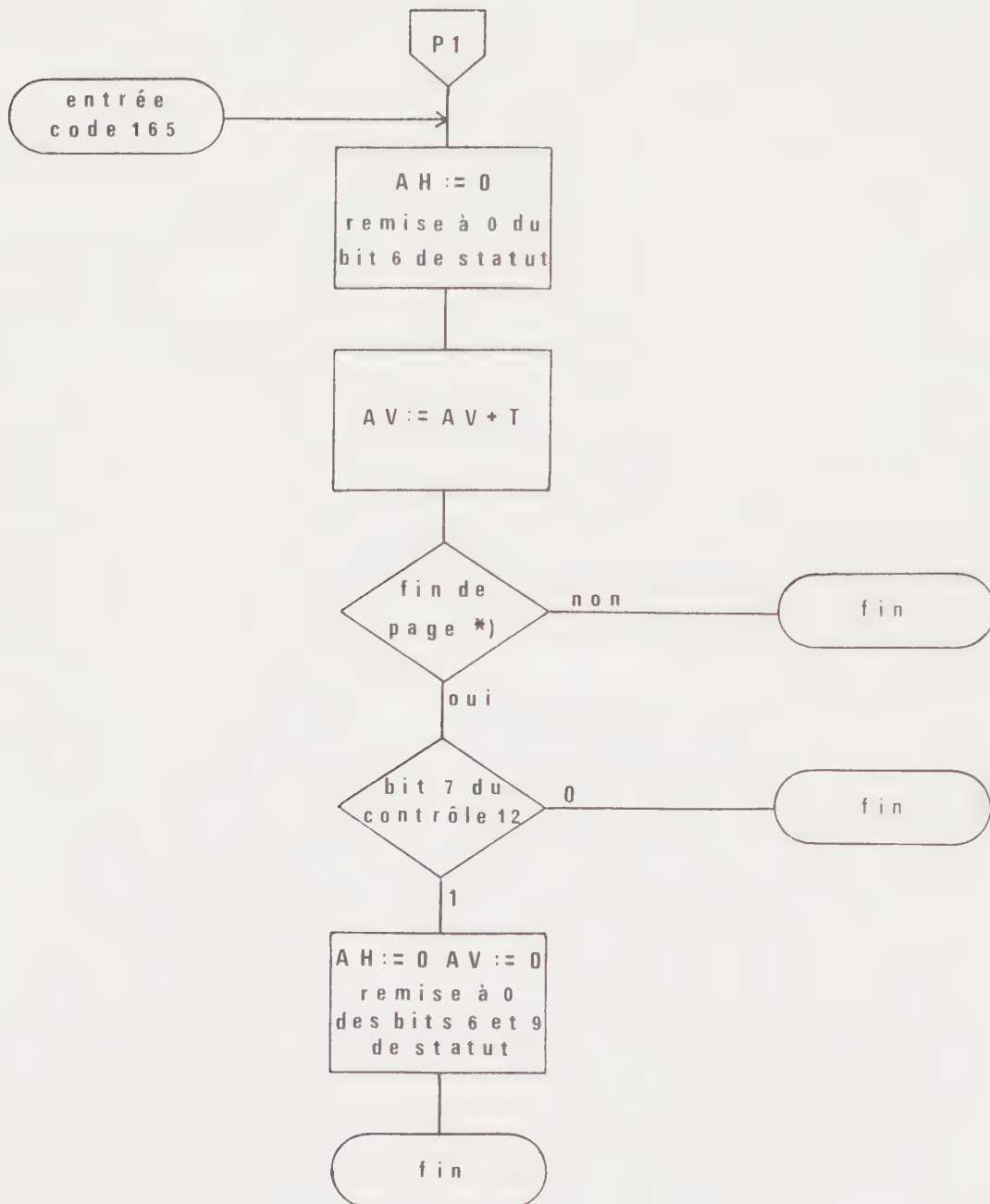
En outre, les adresses peuvent avoir été préalablement fixées par une commande ad hoc. L'adresse de chaque caractère correspond à l'angle supérieur gauche d'un rectangle de 6×8 points.



Dans ces conditions, lors de l'écriture d'un caractère, la taille est ajoutée aux bits les plus significatifs (4 à 11) du registre d'adresse horizontale, selon la valeur du contrôle 12, le dispositif de commande effectue les retours automatiques s'il y a lieu.

Logique des retours automatiques





Abréviations : AH : adresse horizontale
AV : adresse verticale
T : taille

*) Les tests de fin de ligne et de fin de page se font sur l'état des bits de statut 6 et 9 relevés avant le début de l'opération, c'est-à-dire avant les modifications des registres d'adresse. Pour que ces tests aient un sens, il faut que les registres d'adresse soient actifs.

Avec cette logique, une ligne ou une colonne peut contenir

$$\left[\frac{M + 2T - 1}{T} \right]$$

caractères, M désignant la valeur de la marge correspondante. Le nombre de caractères par ligne et de lignes par page se présente donc ainsi :

taille	caractères/ligne	lignes/page
1	241	139
2	121	70
3	81	47
4	61	36
5	49	29
6	41	24
7	36	21

Contrôle 14

Si le bit 0 de ce contrôle auxiliaire vaut 0, un balayage se produit sur la ligne repérée par les registres d'adresse. La largeur de ce balayage est égale à la taille. En mode effacement, ce contrôle est utilisé pour nettoyer une ligne avant de l'écrire.

Contrôle 15 : mises à zéro



- bit 1 : mise à zéro de l'adresse horizontale
- bit 2 : mise à zéro de l'adresse verticale
- bit 3 : mise à zéro des 4 registres de caractère
- bit 4 : mise de la valeur 212₈ dans le registre de marge verticale
- bit 5 : mise de la valeur 360₈ dans le registre de marge horizontale
- bit 6 : mise à zéro du contrôle 12
- bit 7 : mise à zéro du contrôle 13.

7.2 Le driver DRDYD

Ce driver interprète les fonctions STA, WRF, WEF, SDH. Comme dans DRTWD, STA sert de sortie commune à la fin des différentes opérations.

WEF provoque deux sauts de ligne et l'écriture de "\$\$\$".

Comme cette fonction est toujours envoyée au moment de la terminaison d'un travail, l'utilisateur est averti si celle-ci est anormale.

SDH (mise en densité 556 caractères/pouce) n'a évidemment aucun sens sur un écran. Toutefois, cette fonction, envoyée par IOC au moment de l'assignation, est interprétée afin d'effectuer les initialisations et de sélectionner les options par défaut : effacement de l'écran, mise à zéro de tous les registres, sélection des retours automatiques, de la taille et de l'intensité standard (4) et remise en lecture.

Pour l'écriture (WRF), le mode binaire et le mode BCD sont définis : le premier correspond à l'envoi de bytes de 12 bits selon le code défini à la section 7.1, le second à l'écriture de texte selon le code interne.

Une table de conversion permet de passer du code BCD au code ASCII, elle est formée de 8 mots de 8 bytes de 6 bits, le (k + 1)-ème byte contient le code BCD interne du caractère dont le code ASCII est k.

Deux versions de DRDYD ont été écrites.

Dans la première, le mode binaire provoquait le transfert des données directement, à partir de la zone de l'utilisateur.

Pour le mode BCD, le code traduit était stocké dans une zone réservée à l'intérieur du driver, appelée DYBUF, puis envoyé à l'équipement.

Les désavantages étaient :

- Lors d'écritures en grandes quantités, il n'était pas possible de prendre connaissance de la i-ème page, car elle était très rapidement recouverte par la (i + 1)-ème.
- Les écritures de la (i + 1)-ème page étaient superposées sur celles des i précédentes, après 3 pages l'écran était totalement illisible, si l'utilisateur n'insérait pas de commande d'effacement. Une telle commande d'effacement était contraire

à l'un des principes de base de ce travail : les ordres d'entrée-sortie doivent être les mêmes pour tous les périphériques, le rôle des drivers étant précisément de tenir compte de leurs différences.

Il a donc fallu ajouter une possibilité de temporisation à l'arrivée en bas de page, ainsi que l'effacement des lignes avant l'écriture. Deux adjonctions dans le hardware ont été nécessaires : au contrôle 14, libre jusqu'ici, a été attribué l'effacement de la ligne, et une possibilité d'intervention manuelle, liée à une interruption (fonctions 100₈ et 200₈) a été introduite.

Dans sa deuxième version, DRDYD tient compte de ces nouvelles possibilités, mais pour savoir quand les faire intervenir, il doit toujours connaître la position sur l'écran. La routine CTRL, qui calcule les modifications des registres d'adresse et qui introduit au besoin les codes d'effacement, a été ajoutée, elle est décrite dans la section 7.3.

Pour le mode BCD, l'appel à CTRL se fait immédiatement après la conversion de code; en mode binaire, les bytes, pris dans la zone de l'utilisateur, passent également à travers CTRL avant d'être stockés dans DYBUF.

Contrairement à la machine à écrire, pour laquelle il reste une exception, le mode binaire a un sens pour le programmeur utilisant le dispositif de visualisation, puisqu'il lui donne accès à tous les contrôles, ce qui est nécessaire pour des représentations graphiques.

Le mode binaire est accessible en FORTRAN par l'instruction

```
WRITE (i) liste de variables
```

Les routines d'entrée-sortie du FORTRAN préfacent les données par un mot servant de compteur, qui a un sens pour un disque ou pour une bande magnétique, où l'information est relue, mais qui risque de gêner, voire de fausser le déroulement sur le dispositif de visualisation. Lors d'une écriture binaire, DRDYD a la possibilité d'ignorer ou non le premier mot. A cet effet, il utilise les bits de densité : le premier mot n'est transmis que si RHT.DN = 1, c'est-à-dire si une requête

MODE DY,*,LO

a été effectuée au préalable. Sans déclaration particulière (ce qui est le cas du FORTRAN), RHT.DN = 0 (la fonction initiale SDH n'affecte pas RHT.DN), ainsi le problème est résolu.

Le mode BCD travaille avec les caractères de contrôle suivants :

- 1 : dès que l'utilisateur a appuyé sur le bouton d'interruption, l'écran est entièrement effacé et l'écriture se poursuit sur la première ligne;
- 0 : l'écriture est précédée d'un double interligne;
- + : l'écriture recommence au début de la ligne en cours;
- : l'écriture se poursuit à la suite du dernier caractère écrit;
tout autre caractère est ignoré et un passage à la ligne précède l'écriture.

A part le caractère de contrôle -, tous les autres correspondent rigoureusement à ceux de l'imprimante, celui-ci a le même effet que sur la machine à écrire.

D'autres conventions sont analogues à celles de la machine à écrire :

- DRDYD passe de la notion de compte de mots à celle de compte de caractères ou de bytes. Si le compte de mots vaut N, il donne lieu à $8 \cdot N$ caractères en mode BCD et à $4 \cdot N$ bytes en mode binaire.
- En mode binaire les mots de contrôle IOTR et IOTW sont équivalents.
- En mode BCD, le premier caractère n'est considéré comme caractère de contrôle que si le mot de contrôle précédent est du type IOS^T_R .
- Avant le premier mot de contrôle d'une chaîne est toujours inséré un
IOTR,C (0)*,0
- Les mots de contrôle du type IOS^R_W provoquent l'envoi de zéros, leur nombre dépend du mode, et le premier est interprété comme caractère de contrôle en mode BCD.
- Dans ce même mode BCD, les espaces ne sont pas écrits au moment de leur arrivée mais comptés, ils ne sont écrits que

si un caractère différent parvient au driver avant l'épuisement du mot de contrôle.

Un caractère n'ayant aucun effet sur l'écran a été prévu : le code 36₈ est converti en 0400₈ (le contrôle 1 est ignoré). Placé à la suite d'espaces, ce caractère force leur écriture.

Une différence entre l'écran et l'imprimante doit être relevée : la notion de ligne, bien définie sur l'imprimante (136 caractères au maximum, dont un de contrôle) s'avère beaucoup plus variable sur l'écran : le contenu d'une ligne d'imprimante peut donner lieu à plusieurs lignes sur l'écran, en fonction des marges et de la taille. Pour l'utilisateur, cela pose des problèmes de mise en page et de portabilité.

C'est la taille de la ligne sur l'imprimante, et de l'enregistrement BCD en FORTRAN (136 caractères) qui ont déterminé la longueur de DYBUF.

Avec les marges par défaut, en taille 7, un texte de 135 caractères s'étend sur 4 lignes de l'écran, ce qui donne le compte suivant :

éléments	nombre de bytes
mise en écriture et caractère de contrôle	10
135 caractères	135
3 passages à la ligne	15
	160

soit 40 mots. A cela s'ajoutent la remise en lecture et certaines autres conditions spéciales qui demandent 2 mots supplémentaires, non accessibles à l'utilisateur.

Rien n'empêche un programme en assembleur de transmettre plus de 17 mots en BCD, ou encore, le tracé d'une figure (en mode binaire) peut nécessiter plus de 40 mots.

Dans la première version de DRDYD, le nombre de 40 mots dans DYBUF était une limite infranchissable.

La deuxième version n'impose pas cette restriction : elle fractionne l'opération et envoie à l'équipement le contenu de DYBUF autant de fois que nécessaire. Au moment de l'interruption, le driver reprend sans autre l'opération là où il l'avait laissée.

Un autre cas qui entraîne le fractionnement est le passage à une nouvelle page, en mode BCD et pour autant que certaines conditions en relation avec l'emploi du clavier soient remplies (voir sections 7.3 et 8.3). Cette situation provoque l'écriture d'un rectangle plein, de la taille d'un caractère, à côté du dernier caractère affiché, la remise en lecture et la sélection de l'interruption manuelle. L'écriture, au haut de l'écran, ne se poursuit qu'après réception de l'interruption manuelle.

Variables utilisées par DRDYD

Comme dans DRTWD, UPT sert à stocker le nombre de caractères ou de bytes.

DRDYD utilise, en outre, une zone de mots de contrôle, DYOW, ainsi qu'une zone auxiliaire de 7 mots : HADDR, VADDR, HMARGIN, VMARGIN, CONTROL, SIZE, BLCOUNT. Ces mots contiennent respectivement les valeurs des registres d'adresse, de marge, les contrôles 12 et 13, la taille (dans les bits 4 à 6) et le compteur d'espaces.

D'autres valeurs sont stockées dans des mots faisant partie du driver : les pointeurs permettant de reprendre une opération fractionnée ainsi que deux flags indiquant le motif de fractionnement (fin de DYBUF ou fin de page). Ce mode de stockage est possible, car le driver ne peut pas être déplacé, pendant que ces valeurs doivent être sauvegardées.

Utilisation du système d'interruption

Ce point est plus simple pour DRDYD que pour DRTWD, car il n'y a qu'un sens de transfert. Lors d'une écriture, DRDYD sélectionne l'interruption sur unité disponible, appelle XBEGW et sort par DRBSY. Dans le cas cité ci-dessus, l'interruption manuelle est également sélectionnée.

Après réception des interruptions, le traitement se poursuit comme s'il s'agissait d'une entrée d'initialisation. Pour autant que l'interruption manuelle n'intervienne pas, l'écriture du contenu de DYBUF demande donc deux passages à travers les routines traitant les interruptions. L'épuisement du dernier mot de contrôle de la chaîne amène la fin par STA et DRFIN.

7.3 - Contrôle de la mise en page

La routine principale de mise en page est CTRL, elle est appelée, quel que soit le mode, chaque fois qu'un byte est obtenu.

CTRL se compose de plusieurs routines indépendantes, traitant chacune un des contrôles décrits dans la section 7.1.

Chacune de ces routines simule sur les variables HADDR, VADDR, HMARGIN, VMARGIN, CONTROL et SIZE la ou les commande(s) reçues dans chaque byte, mais aucune opération n'est effectuée sur le dispositif de visualisation; en fin de traitement, le byte est stocké dans DYBUF par la routine STORE. Lorsque le mot de contrôle de l'utilisateur est épuisé, ou lorsque la zone DYBUF est pleine, l'écriture a lieu.

Seul le traitement du contrôle 0 et des retours automatiques diffère du comportement hardware. Dans la logique des retours automatiques (voir organigramme de la section 7.1) les retours s'effectuent après l'écriture du caractère. Ceci présente un inconvénient : si un texte comporte exactement le nombre de caractères qu'il est possible d'écrire sur une ligne, le passage à la ligne se fait tout de même.

Dans CTRL, les tests et les retours précèdent le stockage du byte. De cette manière, le nombre de caractères par ligne (ou de lignes par page) vaut

$$\left[\frac{M + T - 1}{T} \right] \text{ au lieu de } \left[\frac{M + 2T - 1}{T} \right] .$$

Lors du traitement d'un byte, si HADDR < HMARGIN; alors HADDR := HADDR + SIZE, et le byte est stocké dans DYBUF.

Si par contre HADDR ≥ HMARGIN, si de plus les registres d'adresse sont actifs et le passage à la ligne automatique est sélectionné dans le contrôle 12, une séquence de retour à la ligne et d'effacement de la ligne est insérée :

0163 6402 6502 7000 6503 .

Puis HADDR := 0 et VADDR := VADDR + SIZE .

La fonction 0163, dans le code binaire de l'utilisateur produit également la même séquence.

Si $VADDR < VMARGIN$, alors $HADDR := HADDR + SIZE$, et le byte est stocké.

Si au contraire, $VADDR \geq VMARGIN$, si de plus les registres d'adresse sont actifs et si le retour automatique en haut de page est sélectionné dans le contrôle 12, le passage à la page suivante a lieu.

En mode BCD, ce traitement se fait en deux phases.

Dans la première, CTRL

- supprime dans DYBUF les 5 bytes de passage à la ligne qui précèdent;
- pose $HADDR := HMARGIN$ (pour forcer une condition de fin de ligne) et $ENDPAGE := 100_8$;
- insère dans DYBUF un rectangle plein de la grandeur d'un caractère, ainsi que la remise en lecture;
- sollicite l'intervention de l'utilisateur en sélectionnant l'interruption manuelle;
- lance l'écriture du contenu de DYBUF et sort à DRBSY.

Ceci laisse à l'utilisateur le temps de prendre connaissance de la page, d'envoyer le signal d'interruption, ce qui provoque la deuxième phase.

CTRL est à nouveau exécuté, les opérations ci-dessus sont répétées, mais cette fois la variable $ENDPAGE \neq 0$, elle est remise à zéro et la séquence d'intervention manuelle n'est plus exécutée. CTRL

- stocke dans DYBUF la séquence de passage en haut de page :
4000 6402 6502 7000 6503 ;
- pose $HADDR := 0$ et $VADDR := 0$;
- procède finalement au stockage du caractère envoyé à CTRL et met à jour $HADDR := HADDR + SIZE$.

Le traitement du caractère de contrôle 1 est analogue : au départ une condition logique de bas de page est forcée, quelle que soit la valeur effective de l'adresse verticale : $VADDR := VMARGIN$, puis CTRL est appelé avec le code 0163_8 . Dans la deuxième phase, le byte d'effacement 6606_8 précède la séquence de passage en haut de page.

La sélection de l'intervention manuelle lors du passage à une nouvelle page dépend encore de l'interaction entre le clavier et l'écran (voir section 8.3). En mode binaire, l'intervention manuelle n'est pas sollicitée pour un passage à une nouvelle page et tout se passe en une seule phase.

En aucun cas, les retours automatiques ne sont effectués par le hardware, même en mode binaire, c'est le traitement software qui est appliqué. Les bits 6 et 7 du contrôle 12 servent à sélectionner ou non cette forme de traitement.

La longueur de DYBUF nécessite encore une précaution : les séquences de passage à la ligne ou à la page ne peuvent pas être interrompues par la fin de cette zone. C'est pourquoi, elle possède deux mots supplémentaires, jamais utilisés dans le traitement usuel, et accessibles seulement à CTRL. La routine STORE veille à ce que ces deux mots ne soient pas dépassés, et si le cas se produit, ce qui ne peut être que la conséquence d'une situation anormale, le travail est abandonné.

Chapitre 8 : Le clavier alphanumérique et de fonctions

8.1 Caractéristiques de l'équipement

L'unité clavier se compose de deux parties bien distinctes : un clavier alphanumérique Honeywell 67SW1-2 et un clavier de fonctions conçu et réalisé par le groupe hardware du CICE. Le clavier de fonctions comporte 64 cases sous forme d'une grille 8×8 . A chaque ligne et à chaque colonne correspond un faisceau lumineux, le fait de couper deux faisceaux, c'est-à-dire de sélectionner une case, place une donnée dans une mémoire tampon de 12 bits, également utilisée par le clavier alphanumérique. En outre, des lampes de sélection, associées à chacune des fonctions, permettent, dans un programme interactif, de guider la réponse de l'utilisateur. Cette forme est extrêmement souple, car la signification attachée à chaque case du clavier de fonctions peut varier d'un programme à l'autre : il n'y a pas de touches gravées une fois pour toutes, il suffit de changer un masque sur une feuille de papier calque.

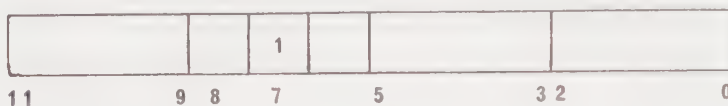
Les codes de fonctions pour l'instruction EXTf sont (en octal) :

0	déconnection	
5	libération de toutes les interruptions et effacement de la mémoire tampon	
20	sélection	} de l'interruption sur
21	libération	
22	sélection	} de l'interruption sur
23	libération	
100	sélection	} de l'interruption sur caractère prêt
200	libération	
		(en provenance du clavier alphanumérique ou de fonctions).

Les bits de statut sont :

- 0 prêt
- 1 occupé
- 3 interruption sur caractère prêt
- 6 statut manuel (un bouton spécial permet de l'enclencher)
- 7 interruption sur unité disponible
- 8 interruption sur fin d'opération.

La seule opération d'écriture est l'allumage des lampes de sélection, elle se fait par des bytes de 12 bits.



bits 0-2 : numéro de la colonne (entre 0 et 7)

bits 3-5 : numéro de la ligne (entre 0 et 7)

bit 7 : toujours à 1

bit 8 : provoque l'extinction de toutes les lampes
s'il vaut 1.

Un byte entièrement nul est sans effet.

C'est par des bytes de 12 bits, également, que se fait la lecture. Le clavier alphanumérique travaille en code ASCII à 7 bits (voir annexe 2), le bit 7 est toujours à 0. Le passage de majuscule en minuscule, ou vice-versa, n'envoie aucun code mais modifie les codes des caractères suivants.

Les codes en provenance du clavier de fonctions sont les mêmes que ceux utilisés pour l'allumage des lampes de sélection.

Donc, seul le bit 7 permet de distinguer la provenance d'un byte envoyé par l'équipement.

Lors d'une lecture, un seul byte de 12 bits est placé dans un mot, le reste est complété par des zéros.

De même que sur la machine à écrire, une lecture peut se faire de deux façons. La première consiste à sélectionner l'interruption sur caractère prêt, et lors de son arrivée à effectuer la lecture du code. La seconde n'a recours qu'à l'instruction machine BEGR et bloque le canal jusqu'à épuisement du mot de contrôle.

Le fait de sélectionner simultanément deux caractères sur le clavier alphanumérique, deux faisceaux sur le clavier de fonctions ou un caractère et une fonction est une cause d'erreur, un signal acoustique se fait entendre, l'utilisateur doit appuyer sur un bouton pour effacer l'erreur, le caractère fautif est ignoré.

8.2 Le driver DRKBD

Ce driver traite les fonctions RDF, WRF, STA et RDY.

STA et RDY se ramènent à une simple prise de statut et constituent la sortie du driver.

Pour la lecture et l'écriture, les deux modes, binaire et BCD sont interprétés.

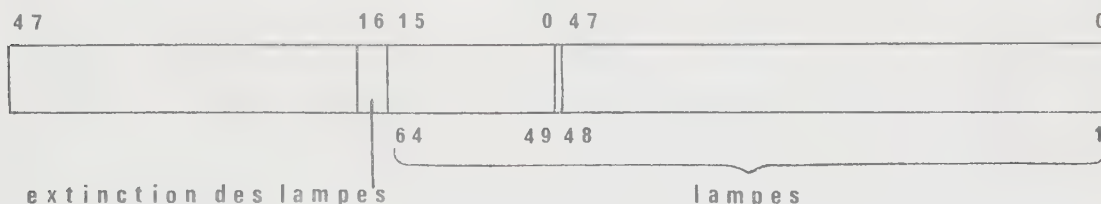
Ecriture

L'écriture binaire (allumage des lampes de sélection) correspond à l'envoi de bytes de 12 bits dans le code du clavier de fonctions. La convention au sujet du premier mot est la même que dans DRDYD : il n'est transmis que si une requête

MODE KB,*,LO

a été effectuée auparavant, sinon il est ignoré.

L'écriture BCD permet de coder les sélections sur deux mots :



A chaque lampe, ainsi qu'à l'extinction préalable de toutes les lampes correspond un bit; pour les lampes, la valeur 1 signifie allumage, la valeur 0 ne provoque pas de changement; si le bit 16 du premier mot vaut 1, l'extinction préalable a lieu.

Ce mode d'écriture est utilisable sans autre en assembleur, en FORTRAN, il faut que les deux mots soient transmis sans conversion, c'est-à-dire à l'aide d'un

FORMAT(2A8) .

Dans ce mode, le driver fixe toujours le compte de mots à 2 et effectue le décodage. L'allumage de chaque lampe fait l'objet d'une écriture physique, tandis qu'en mode binaire, un mot de contrôle dirige l'envoi de tous les codes.

Interprétation des mots de contrôle en écriture

Dans les deux modes, les mots de contrôle IOTR et IOTW sont traités de la même manière et les chaînages sont effectués. Les mots de contrôle IOS_W^R ont une interprétation tout à fait spéciale. En lecture, l'utilisateur a la possibilité de se définir une table de mots-clés associés aux différentes fonctions de son programme. L'intérêt de ces mots-clés est qu'ils sont rendus équivalents à des séquences de caractères tapés sur le clavier alphanumérique. Chaque mot-clé peut compter un ou plusieurs mots machine, par convention, ils doivent tous en avoir le même nombre (usuellement : 1 ou 2). Soit une table de N ($N \leq 64$) mots-clés de M mots machine chacun. Elle doit se présenter en mémoire de la manière suivante :

1er mot-clé										
2ème mot-clé										
Nème mot-clé										
	1	2								M

Ce qui caractérise cette table, c'est son origine et le nombre de mots machine par mot-clé, cela suffit pour retrouver n'importe quel mot-clé.

Par convention, une requête WRITE, pour laquelle le mot de contrôle est IOS_W^R sert à transmettre à DRKBD ces deux informations.

Exemple :

```

                WRITE      KB,CWT,*
                :
                :
CWT      IOSR      (*)T,3

```

La table commence à l'adresse T, chaque mot-clé comporte 3 mots machine.

Dans ce cas, la seule action de DRKBD est de stocker le mot de contrôle dans une position de mémoire réservée. Si le compte est nul, le driver considère qu'il n'y a pas de table.

Lecture

En lecture, le traitement des bytes reçus dépend essentiellement de leur provenance. Seule l'initialisation est constante. Les données sont stockées au fur et à mesure de leur arrivée dans la zone de l'utilisateur, converties ou non. En début d'opération, celle-ci est remplie par des zéros en mode binaire et par des espaces en mode BCD.

Le compte de mots (N) est remplacé en mode binaire par le compte de bytes de 12 bits ($4 \cdot N$) et en mode BCD par le compte de caractères de 6 bits ($8 \cdot N$).

Lecture sur le clavier alphanumérique

Lors d'une lecture en mode binaire, des bytes de 12 bits en code ASCII sont transmis; en mode BCD, ce sont des caractères en code interne. Le code ASCII comporte plusieurs codes spéciaux qui ne peuvent avoir de correspondant dans un code à 64 caractères. A certains de ces codes figurant sur le clavier alphanumérique une signification particulière a été attribuée de manière plus ou moins arbitraire (voir également section 8.3). Dans les deux modes, trois touches ont un rôle privilégié :

- La touche CR marque la fin du mot de contrôle en cours d'exécution.
- La touche BS annule le dernier byte et le remplace par un byte nul en mode binaire, elle annule le dernier caractère et le remplace par un espace en mode BCD.

- La touche BEL provoque l'annulation de tout ce qui a déjà été transmis par le mot de contrôle en cours, ainsi que la remise dans l'état initial de la zone de l'utilisateur.

Pour la conversion de code ASCII en code BCD interne, une table de 12 mots de 8 bytes de 6 bits est utilisée. Le k-ième byte donne le code BCD du caractère dont le code ASCII est $k + 37_8$. Tous les caractères alphabétiques ont deux codes différents, l'un correspond à la position majuscule, l'autre à la position minuscule du clavier. Les codes inférieurs à 40_8 correspondent à des caractères spéciaux et ne sont pas transmis à l'utilisateur.

Lecture sur le clavier de fonctions

Dans le cas général, le clavier de fonctions ne peut se suffire à lui-même, il doit être considéré comme un complément au clavier alphanumérique avec lequel il ne constitue, d'ailleurs, qu'une seule unité.

En mode BCD, deux types d'utilisation ont été prévus :

- L'utilisateur travaille essentiellement avec le clavier de fonctions et il ne donne pas de table de mots-clés (ou il en indique une de longueur nulle, ce qui revient au même). Dans ce cas, DRKBD envoie à chaque sélection de case un code de 6 bits, équivalent au code envoyé par le clavier, mais dont le bit 7 a été remis à zéro. Ce code est donc un nombre entre 0 et 63, pour un langage tel que le FORTRAN, il doit être lu selon un

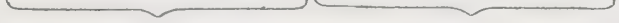
FORMAT(A8)

(lecture sans conversion). Au cas où le compte de mots est supérieur à 1, ce code est stocké dans le premier mot de la zone de lecture. Chaque sélection de case amène la fin de l'opération, le contrôle est donc immédiatement rendu au programme de l'utilisateur qui doit tester puis traiter le code transmis. Avec ces conventions, les codes de correction du clavier alphanumérique sont inapplicables, c'est à l'utilisateur de prévoir une case signifiant l'annulation du code transmis lors de l'opération précédente.

- Le deuxième type d'utilisation a recours aux mots-clés. Grâce à eux, les possibilités d'utilisation conjointe sont poussées plus loin. Chaque mot-clé correspond donc à une suite bien définie de caractères et à une case du clavier de fonctions. Pour plus de commodité, l'utilisateur peut écrire chaque mot-clé sur un papier calque à l'emplacement de la case correspondante et glisser ce papier sous la grille amovible du clavier de fonctions. Au cours d'une opération de lecture, on peut passer plusieurs fois successivement d'un clavier à l'autre, cette lecture se termine par le code CR du clavier alphanumérique (ou par l'épuisement du compte de mots), d'autre part les touches de correction sont utilisables.

Exemple :

Le langage INFOL (section 9.5) travaille avec des mots-clés de 2 mots machine. Le message suivant peut donc être transmis en 4 mouvements du doigt :

ESTABLISHMENT	INITIAL INPUT	5	1 touche +
			1 touche CR
1 case du clavier de fonctions	1 case du clavier de fonctions		

Sur le clavier alphanumérique, il faut appuyer successivement sur 34 touches pour obtenir un message identique. Ce type d'utilisation est intéressant à cause de sa souplesse et aussi parce qu'il permet d'éviter bien des erreurs de frappe. Il faut cependant que l'utilisateur prévoie dans son programme la lecture en format libre, surtout afin d'éliminer les espaces superflus; ceci n'est pas une contrainte propre au clavier de fonctions, on la retrouve avec n'importe quelle utilisation d'un clavier ou d'une machine à écrire.

Pour une bonne utilisation interactive, le recours aux lampes de sélection est judicieux, au fur et à mesure du déroulement du programme, les lampes correspondant aux réponses possibles attendues peuvent être allumées.

Le clavier de fonctions peut également travailler en mode binaire : tout se comporte comme en mode BCD sans mots-clés, mais dans ce cas le bit 7 du code, valant 1, est transmis.

Interprétation des mots de contrôle en lecture

Les mots de contrôle IOTR et IOTW sont équivalents, leur exécution se termine par la réalisation d'une des 3 conditions :

- sélection d'une case sur le clavier de fonctions en l'absence de mots-clés;
- envoi d'un CR sur le clavier alphanumérique;
- retour à zéro du compte de mots.

DRKBD effectue les chaînages et lorsque le dernier mot de contrôle de la chaîne est épuisé, il sort à DRFIN par STA.

Les mots de contrôle IOS_W^R demandent une lecture mais aucune transmission n'a lieu hors du driver.

Mot de contrôle de fin d'opération

Comme dans DRTWD (voir section 6.2) le mot de contrôle de fin d'opération est calculé à partir de la chaîne de l'utilisateur. Pour éviter les incompatibilités avec les routines du FORTRAN, le compte de mots est ramené à 0, quelle que soit la longueur de l'information transmise.

Utilisation du système d'interruption

L'allumage des lampes de sélection ne prend que quelques μs , l'écriture se fait par SBEGW sans intervention du système d'interruption. Une boucle d'attente testant le bit de statut 13 permet de déterminer la fin du transfert. En mode BCD, une boucle d'attente a lieu pour chaque allumage, en mode binaire, la transmission de tous les bytes en une fois ne nécessite qu'une seule boucle d'attente.

Pour la lecture, DRKBD sélectionne l'interruption sur caractère prêt et sort par DRBSY. A la réception de cette interruption, la lecture est effectuée par SBEGR, une boucle testant le bit de statut 12 attend la fin de l'opération. Après conversion et transmission du code reçu (pour autant qu'elles soient nécessaires), DRKBD passe au byte ou au mot de contrôle suivant, selon les cas. La lecture d'un byte ne demande donc qu'un passage à travers les routines gérant le système d'interruption.

De même que pour DRTWD, un paramètre d'assemblage offre la possibilité de faire la lecture sous le contrôle de XBEGR, ce qui demande deux interruptions supplémentaires; cette option n'a pas été retenue pour la version standard.

Relation entre le clavier et l'écran de visualisation

En mode BCD, toutes les informations en provenance des deux claviers sont affichées simultanément sur l'écran, à la suite de ce qui y figure déjà. En outre, par un jeu de commandes souple, l'utilisateur peut diriger la mise en page sur l'écran. Ces commandes ne peuvent être passées qu'avec un message demandé par le programme, elles n'ont aucun effet sur le texte transmis.

Le clavier et l'écran étant deux unités distinctes et aucune liaison physique n'existant entre eux, toutes ces opérations sont programmées.

Un appel à DRDYD est donc nécessaire, or l'appel de driver à driver n'est pas possible. La solution consiste à faire un appel indirect : DRKBD demande à la routine ISINL (dans INC) de faire en sorte que le programme 63 donne le contrôle à une petite routine résidente, KBDYR, servant d'amorce à l'overlay KBDYL. Après l'appel à ISINL, DRKBD sort par DRBSY bien qu'aucune interruption ne soit attendue sur le clavier, afin que le contrôle puisse revenir au niveau 63 et à KBDYR en particulier. Celui-ci appelle l'overlay (voir annexe 3) KBDYL, également exécuté au niveau 63 et qui contient les requêtes d'écriture provoquant les appels à DRDYD. Finalement KBDYR rappelle DRKBD pour qu'il puisse lire le caractère suivant, s'il y a lieu, et sinon sortir par DRFIN et rendre le contrôle au programme utilisant le clavier. Les détails de tout ce processus font l'objet de la section 8.3.

La transmission des paramètres de DRKBD à KBDYR est un élément essentiel, ils sont stockés dans un mot, de la manière suivante :



bit 47 : une opération est à effectuer sur l'écran.

bits 39-41 : code identifiant le type d'appel :

000 : caractère spécial qui n'a pas d'équivalent BCD,

001 : caractère ayant un correspondant BCD,

010 : fonction (une table de mots-clés est présente),

100 : fonction (il n'y a pas de table de mots-clés).

bits 24-38 : LUD du clavier.

bits 0-14 : leur signification dépend des bits 39-41.

bits 39-41

bits 0-14

000

code du byte en ASCII,

001

code du caractère en BCD interne,

010

déplacement par rapport à l'origine de la table des mots-clés,

100

numéro de la fonction (code transmis par le clavier, sans le bit 7).

Variables utilisées par DRKBD

Comme dans les deux autres drivers, le compte de caractères ou de bytes est stocké dans UPT.

DRKBD utilise une zone de mots de contrôle, KBCW, ainsi qu'une zone auxiliaire, KBAUX, de 3 mots :

- TABLE contient les informations relatives à la table de mots-clés;
- CODE est le mot contenant les paramètres à KBDYL;
- INDX sert à stocker les registres effectuant l'indiaçage dans la zone de l'utilisateur, le bit 47 de ce mot joue un rôle particulier (voir section 8.3).

8.3 L'overlay KBDYL

L'ensemble clavier-écran pouvant être utilisé par un programme central ou par un programme background, il s'ensuit que la routine d'affichage simultané doit se trouver au niveau 63. La présence de cette routine dans le code résident constituerait une perte de place, compte tenu de son faible taux d'utilisation. La meilleure solution consiste à en faire un overlay, KBDYL, en ne laissant résidente qu'une petite portion, KBDYR, qui commande le chargement de l'overlay et lui transmet le contrôle.

En tant qu'overlay, KBDYL est traité par PREDRUM dans la phase d'initialisation du système, il a accès à tous les points d'entrée de celui-ci, ainsi qu'aux zones DY AUX et KBAUX.

Le programme 63 dispose de quelques unités logiques : LIB (la bibliothèque), ICM (la machine à écrire de la console considérée comme moyen d'entrée), ACC (l'unité sur laquelle est écrite la comptabilité), etc. Pour l'affichage simultané, une autre unité lui a été ajoutée : DYS. Ainsi, le dispositif de visualisation appartient à la fois au niveau 63 pour l'affichage simultané et, pour les autres opérations, au programme central ou background utilisant l'ensemble clavier-écran. Le fait qu'une même unité physique appartienne à deux unités logiques différentes ne constitue pas un cas particulier : il en va de même pour les disques et la machine à écrire de la console.

KBDYL commence par des recherches dans les tables. La valeur du LUD du clavier, transmise dans CODE permet de déduire le niveau du programme. Pour que l'écriture sur l'écran puisse avoir lieu, il faut que les recherches suivantes soient effectuées avec succès :

- Un équipement doit porter le mnémonique DY dans HTL, son ordinal dans cette table fournit le numéro du type d'équipement.
- A partir de PSL, puis par PCT, KBDYL détermine l'adresse de la première position de LUB. Une entrée RHT doit contenir dans le champ RHT.H le numéro d'équipement du dispositif de visualisation.

- Le champ RHT.K doit être non nul, c'est-à-dire l'écran doit déjà être assigné.
- Ce champ, contenant l'ordinal AET, donne accès à l'entrée AET de l'écran, le bit AET.D doit être à 0 (l'unité doit être en état de marche).
- Le bit USTQ.Y doit également être nul (l'écran ne doit pas faire l'objet d'un bypass).

Si l'une de ces conditions n'est pas remplie, la séquence de sortie de KBDYL est appelée.

KBDYL se divise ensuite en quatre portions traitant chacune l'un des cas possibles selon la valeur des bits 39-41 de CODE. Ce traitement consiste à préparer l'information qui va être affichée sur l'écran, ainsi que le mot de contrôle qui lui est lié.

Dans le cas d'un caractère en provenance du clavier alphanumérique, la représentation BCD interne est extraite de CODE, KBDYL la fait précéder du caractère de contrôle - (écriture à la suite) et suivre de codes 36, (les espaces doivent être effectués).

Si l'information provient du clavier de fonctions, l'adresse du mot-clé est calculée à partir des cellules TABLE (adresse de base) et CODE (déplacement). Si la table de mots-clés est absente, c'est le numéro de la case qui est préparé. Dans ces deux cas, KBDYL insère le caractère de contrôle - afin d'avoir un affichage sur la même ligne.

Le dernier cas est celui des caractères spéciaux du clavier alphanumérique.

Certains sont ignorés, d'autres ont un effet exclusivement sur l'écran, mais ne sont pas transmis à l'utilisateur par DRKBD.

Les fonctions sur l'écran sont les suivantes :

- passage à la ligne (LF) (ne produit pas de fin de message);
- retour au haut de l'écran (SO);
- effacement de tout l'écran et retour au haut de celui-ci (FF);
- effacement du bas de l'écran (VT) (en-dessous de la position actuelle du registre d'adresse verticale);

- modification de la taille (ENQ), cette fonction requiert deux commandes : la sélection de la modification (ENQ) et l'indication de la nouvelle taille qui doit être un chiffre entre 1 et 7. Lorsque cette sélection est effectuée, KBDYL demande à DRKBD, en mettant le bit 47 de INDX à 1, de lui transmettre directement les caractères (sans les envoyer à l'utilisateur), le processus habituel reprend dès que KBDYL a reçu un chiffre entre 1 et 7 ou un BS (effacement de la sélection);
- modification de l'intensité (ACK) : le processus est rigoureusement identique;
- correction du dernier caractère (BS) : cette fonction produit son effacement sur l'écran et son remplacement par un espace dans la zone de l'utilisateur;
- correction du dernier enregistrement (BEL) : cette fonction efface sur l'écran la ligne en cours d'écriture et dans la zone de l'utilisateur tout ce qui a déjà été transmis par le mot de contrôle en cours.

L'assignation de DYS se fait par simple transfert des tables du programme utilisant le clavier dans celles de DYS. Seules RHT et USTQ sont transmises avec les modifications dans les champs RHT.DN, RHT.BCD, USTQ.F (dans lequel est placée la fonction WRF) et USTQ.CWA.

KBDYL définit également une adresse d'interruption : celle-ci se trouve dans KBDYR, elle ne saurait être située dans un overlay dont la présence est, par définition, temporaire. Enfin, KBDYL effectue une requête STARTUP sur l'unité DYS et rend le contrôle à KBDYR qui contient une requête RETURN. Ainsi d'autres tâches au niveau 63 ou à des niveaux inférieurs peuvent avoir lieu pendant l'écriture sur le dispositif de visualisation.

A la réception de l'interruption, KBDYR fait une nouvelle requête STARTUP, avec pour paramètre le LUD du clavier, cette entrée dans DRKBD marque la fin effective du traitement du code qui vient d'être lu et affiché. Ce procédé peut sembler insolite : DRKBD avait été quitté par DRBSY et le contrôle lui revient non pas par ICHAN et DRIVERL, mais par STARTUP et DRIVERU, et ceci à la suite d'une interruption sur un autre

équipement. Toutefois, ce procédé n'est pas en désaccord avec la philosophie du système et ne pose pas non plus de problème dans DRKBD. En effet, KBDYL a préalablement remis à zéro la variable CODE, ainsi DRKBD sait qu'il peut demander le caractère suivant et ressortir par DRBSY ou alors terminer l'opération et sortir par DRFIN. Dans l'un et l'autre cas, IPASS2 redonne ensuite le contrôle à KBDYR, celui-ci le rend immédiatement par une requête RETURN.

L'interaction entre le clavier et le retour en haut de page sur l'écran peut être précisée maintenant. L'intervention de l'utilisateur n'est pas requise pendant un affichage simultané (DRDYD teste la valeur de ILUD), ni immédiatement après. Cette dernière interaction est régie par un flag (KBDY dans la zone DY AUX) mis à 0 par DRDYD après chaque écriture et à 1 à chaque sortie de KBDYR.

Il convient de souligner qu'il n'existe aucun lien physique entre l'écran et le clavier. La liaison est purement logique. L'intérêt de cette solution réside dans sa souplesse, il est très facile d'ajouter une nouvelle fonction, cela n'entraînerait aucune modification des drivers, seul KBDYL devrait être complété.

Par contre, ce procédé a le désavantage de nécessiter un grand nombre de requêtes (parfois même une lecture sur le disque si KBDYL n'est pas en mémoire) pour l'affichage de quelques caractères ou de quelques mots. Cela prend évidemment du temps machine, mais à la vitesse humaine, le délai d'attente reste imperceptible.

Chapitre 9 : Utilisation des équipements interactifs

9.1 Evaluation des performances

Les drivers doivent répondre à des exigences de place en mémoire, de temps, de sécurité et de commodité pour l'utilisateur.

Place en mémoire

Un driver fréquemment utilisé peut être déclaré résident, ceux des équipements du terminal n'ont pas besoin de l'être. En principe, ils ne sont chargés en mémoire que lorsqu'un programme fait appel à eux.

Une zone de mémoire, appelée BOVER, doit être réservée en permanence pour le chargement temporaire d'un driver, sur une commande de l'opérateur, pour des contrôles ou des opérations simples. Pour les équipements du terminal, seule la prise de statut a un sens. La longueur de BOVER est fonction de celle du plus grand driver non résident, DRDYD en l'occurrence.

Donc, même absent, DRDYD réserve une portion de mémoire.

Les drivers des équipements du terminal sont notablement plus complexes que la plupart de ceux fournis par le constructeur. Les drivers DRTWD, DRDYD et DRKBD optimisent en premier lieu la place en mémoire et ensuite le temps de calcul. A titre de comparaison, les différents drivers ont les longueurs suivantes :

DRMTD	(bandes magnétiques, version avec algorithmes de récupération d'erreurs)	461
DRMTD	(idem mais sans récupération d'erreurs)	161
DRDRD	(disques utilisés en accès direct)	91
DRCTD	(machine à écrire de la console)	197
DRCRD	(lecteur de cartes)	83

DRCPD	(perforateur de cartes)	67
DRLPD	(imprimante)	154
DRTWD		188
DRDYD		316
DRKBD		196

La zone BOVER passe de 289 à 456 mots. A cela s'ajoutent les nouvelles entrées dans les tables AET, HTL et DEP, les zones des mots de contrôle, les positions auxiliaires et KBDYR. De plus, pour chaque driver et pour KBDYL, il faut compter encore un mot dans les tables indiquant leur position respective sur disque. Enfin pour chaque unité, il faut ajouter encore 4 mots dans les tables EERn (voir section 10.3). Même si le terminal n'est pas utilisé, son adjonction coûte en permanence 233 mots supplémentaires, ce qui est admissible dans une mémoire de 64K.

Temps

Il y a deux aspects à considérer : le temps que prend une opération pour son exécution complète et le temps que cette même opération prend dans l'unité de calcul. D'une manière générale, ce temps-ci est difficile à mesurer, pour deux raisons :

- Lors d'une requête, le système peut entreprendre d'autres opérations qui n'ont rien à voir avec celle-ci; ainsi certaines tâches sont effectuées toutes les 5 ou 10 secondes : certains périphériques sont contrôlés périodiquement. La probabilité d'avoir un biais n'est donc pas négligeable.
- L'horloge de l'ordinateur ne donne que la milliseconde et une précision 10 fois supérieure serait nécessaire. La répétition d'opérations s'impose, mais alors, le risque de biais augmente.

Pour les mesures suivantes, le temps initial est pris juste avant une requête, le temps final à l'entrée de la routine d'interruption. Si on relève le temps de l'horloge, la différence indique le temps nécessaire à l'exécution complète. Si, par contre, on relève le compteur de temps relatif au programme, la différence donne le temps pendant lequel l'unité de calcul a travaillé pour le programme.

Pour la mesure de temps très courts, la meilleure solution consiste à avoir recours aux tables donnant la durée de chaque instruction.

Mesure des temps d'unité de calcul

Une requête d'entrée-sortie que le driver n'est pas capable d'interpréter prend 1,7 ms : c'est le temps passé dans INC, REQUEST, IOC, IOEX, le driver, DRFIN et IPASS2.

Une requête d'entrée-sortie, pour laquelle le compte de mots est nul, dure 2 ms, dans ce cas, le driver fait en plus appel à XCONN, MOV CW et XREL avant de sortir par DRFIN; le périphérique n'effectue aucune opération.

Le traitement d'une interruption dans INC dure environ 500 μ s. Le traitement d'un caractère en mode BCD dans DRTWD ou DRKBD (conversion, appel aux routines d'entrée-sortie du système) prend environ 300 μ s. Ainsi, en plus du traitement de la requête proprement dite, chaque caractère lu sur la machine à écrire ou sur le clavier demande 800 μ s à l'unité de calcul (1 interruption) et chaque caractère écrit par la machine à écrire demande 1300 μ s (2 interruptions).

L'allumage des lampes de sélection s'effectue sous le contrôle de boucles d'attente, sans intervention du système d'interruption : ainsi, le temps total nécessaire à l'allumage de la moitié des lampes du clavier de fonctions est de 3 ms.

Pour l'écriture sur l'écran, en mode BCD, le traitement de chaque caractère est allongé par le passage dans la routine CTRL qui prend en moyenne 125 μ s, ainsi, le temps total nécessaire à un caractère est proche de 200 μ s dans DRDYD, par contre, le temps d'appel aux routines d'entrée-sortie du système et celui du traitement des interruptions sont à répartir sur l'ensemble de l'opération.

Mesure des temps relatifs à l'exécution complète

Cette mesure se fait assez aisément par le procédé décrit ci-dessus.

Pour les opérations de lecture, elle ne donne que des indications sur l'habileté dactylographique de l'utilisateur. Seules

les opérations d'écriture sur la machine à écrire et sur le dispositif de visualisation présentent de l'intérêt.

La vitesse maximale de la machine à écrire est de 63 ms/caractère, soit 15,9 caractères/s. Pratiquement, il faut tenir compte, pour établir une moyenne, du retour de chariot qui précède chaque ligne. Tout compris, l'écriture d'un caractère dure 69 ms, ce qui représente 14,5 caractères/s (ces mesures ont été effectuées par un programme travaillant hors du contrôle du système d'exploitation).

Pour le dispositif de visualisation, il est intéressant de comparer le temps nécessaire à une opération sous DISK SCOPE avec celui nécessaire à la même opération, mesuré sur un programme chargé par le processus d'autoload à partir d'une bande magnétique et travaillant hors du contrôle de DISK SCOPE. Pour ces mesures de temps, la partie électronique, utilisant le réseau pour se synchroniser, joue un rôle important. Les changements de mode dans le contrôle 13 peuvent avoir un délai de 20 ms au maximum. L'écriture des caractères est rapide, elle dépend du nombre de points allumés, le temps de 150 μ s constitue une moyenne.

Si cette écriture peut s'effectuer entièrement dans un délai de 20 ms, elle ne provoque aucun allongement de la durée, c'est pratiquement toujours le cas.

Le temps physique d'écriture dépend donc essentiellement du nombre de séquences de retour à la ligne (contrôles 13 et 14), la croissance se fait selon des paliers.

Ecriture de mots en taille 4 par un programme indépendant de DISK SCOPE :	Temps mesurés (en ms) :
---	-------------------------

Séquence de passage à la ligne (avec effacement) et de mise en écriture :	55
---	----

Séquence de remise en lecture :	38
---------------------------------	----

Ecriture de 1 à 7 mots (passage à la ligne, écriture, remise en lecture) :	95
--	----

Ecriture de 8 à 15 mots (idem, mais avec 2 passages à la ligne) :	154
---	-----

Ecriture de 16 à 18 mots (idem, mais avec 3 passages à la ligne) :	212
--	-----

Ces chiffres sont des moyennes de moyennes de groupes de 10 mesures, la dispersion est de ± 4 ms.

Les mêmes tests ont été effectués sous le contrôle du driver. Il faut tenir compte du temps passé dans CTRL qui s'élève à environ 1 ms par mot. Les discontinuités observées dans le tableau ci-dessus subsistent, mais, entre ces discontinuités, on a une croissance au lieu d'une valeur constante.

Ecriture de mots en taille 4 sous le Temps mesurés (en ms) :
contrôle de DISK SCOPE et de DRDYD :

Ecriture de 1 mot :	95
Ecriture de 7 mots :	104
Ecriture de 8 mots :	165
Ecriture de 15 mots :	170
Ecriture de 16 mots :	231
Ecriture de 18 mots :	233
Ecriture de 19 mots :	272

Ces mesures ont été effectuées de la même manière que celles du premier groupe, la dispersion est de ± 3 ms. Au-delà de 18 mots, il y a un fractionnement pour vider DYBUF, c'est ce qui explique le saut entre les temps d'écriture pour 18 et 19 mots. Pour l'affichage simultané du clavier à l'écran, si KBDYL est en mémoire, le temps d'affichage est celui d'un mot, soit environ 95 ms, ce qui est suffisant pour donner une impression de simultanéité. Ce même affichage prend environ 3,5 ms d'unité de calcul. Les opérations dont la durée est perceptible à l'oeil humain sont les effacements : ainsi l'effacement de toute la mémoire électrostatique dure environ 350 ms.

Si KBDYL n'est pas en mémoire, il doit être chargé; le temps nécessaire à l'affichage est augmenté de celui de l'accès au disque qui est en moyenne de 125 ms. Ce temps est également perdu pour l'unité de calcul qui ne peut rien faire d'autre, sinon attendre la fin du chargement.

Des améliorations sont envisagées au chapitre 10.

Sécurité

Il est évident que les drivers ne doivent en aucun cas détruire une partie du système d'exploitation. Ce point est relativement facile à tester sous le système de simulation.

Les drivers ne doivent pas non plus bloquer le système en mode d'interruption ou provoquer des appels à DISASTER. Ils doivent être capables de se comporter correctement lorsqu'ils sont appelés dans des conditions anormales, par IOERR en particulier. Il est difficile de simuler de telles conditions ainsi que d'imaginer toutes les anomalies qui pourraient se produire. Chaque driver dispose d'un moyen de récupération simple mais efficace : en cas de doute, il efface toutes les interruptions et passe au traitement du caractère suivant. Des récupérations complémentaires ont été introduites dans XCONN et XEXTF pour améliorer le traitement des rejets.

Le comportement des drivers doit également être correct si pour une raison ou une autre DISASTER est appelé. Dans ce cas, le travail en cours est perdu, le travail suivant peut démarrer dans des conditions normales : toutes les zones auxiliaires sont rechargées à partir du disque avec la valeur 0.

Commodité pour l'utilisateur

L'utilisation des équipements du terminal doit être compatible avec celle des équipements déjà existants, ils doivent d'autre part offrir des possibilités supplémentaires, liées à leur nature.

Les jeux de caractères posent un premier problème : il s'agit de concilier le code BCD interne, le code Selectric et les codes ASCII à 6 et 7 bits. Dans les tables de conversion, les lettres, les chiffres et les caractères spéciaux les plus fréquents correspondent, des divergences apparaissent toutefois entre les autres caractères spéciaux (voir annexe 2).

Le problème des caractères de contrôle est également un obstacle sérieux à la portabilité (les options diffèrent déjà d'un constructeur à l'autre), les équipements du terminal n'ont pas pu reprendre à leur compte toutes les options de l'imprimante.

Le nombre de caractères/ligne et celui de lignes/page varient également d'un périphérique à l'autre.

En lecture, la machine à écrire peut recevoir 100 caractères par ligne, sa capacité est donc supérieure à celle d'une carte perforée. Le clavier n'est pas astreint à cette limite et peut envoyer des messages de n'importe quelle longueur.

En écriture, la machine à écrire offre également 100 caractères par ligne, soit moins qu'une imprimante. Les restrictions du dispositif de visualisation sont plus gênantes. La taille 4 n'offre que 60 caractères par ligne et 35 lignes/page, ce qui ne suffit pas. Ces chiffres pourront passer à 80 et 46 quand des améliorations de la partie électronique permettront d'utiliser de manière standard la taille 3.

9.2 Utilisation par un programme background

Les programmes background doivent être écrits en assembleur et stockés en permanence sur le disque, l'introduction de nouveaux programmes background se fait par la bande système. Ils sont prévus surtout pour des transferts de données d'une unité à une autre et ne doivent utiliser qu'un faible temps de calcul.

Ces programmes constituent le milieu optimal pour l'utilisation des équipements du terminal : un programme central peut se dérouler pendant que l'utilisateur tape ses données sur le clavier ou la machine à écrire. Les programmes background offrent ainsi une possibilité restreinte de multiprogrammation.

Pour commencer un tel travail, il faut que le programme background, s'il est non résident ait été auparavant chargé en mémoire centrale par une commande de l'opérateur. Le terminal n'a pas la possibilité d'alerter le système par une interruption, car DISK SCOPE n'admet que des interruptions attendues. Un autre moyen consiste à travailler avec un programme background résident testant périodiquement (toutes les 10 secondes environ) le bit de statut manuel de la machine à écrire ou du clavier. Le bouton associé à ce bit peut alors jouer le rôle d'un bouton d'appel, en le pressant, l'utilisateur signale qu'il désire transmettre quelque chose et le programme entre dans sa phase active.

Quelques programmes background simples ont été écrits et servent à tester le bon fonctionnement des équipements.

Un programme background lit des messages sur la machine à écrire et les y répète, il est capable de travailler en mode binaire ou BCD.

Un autre programme lit des messages sur la machine à écrire, les copie sur le disque d'où un autre programme background les relit et les perfore sur cartes.

Un autre programme lit des messages sur le clavier et les affiche sur l'écran, il offre également la possibilité de travailler dans les deux modes et de contrôler le fonctionnement de toutes les commandes du dispositif de visualisation.

Une autre possibilité consiste à écrire un programme background lisant des travaux à partir du clavier et les intercalant dans la queue d'attente.

9.3 Utilisation par un programme central

Un programme central a accès aux équipements du terminal, pour autant qu'ils ne soient pas à ce moment là sous le contrôle d'un programme background.

Les nombreux temps d'attente sont le désavantage de ce type d'utilisation, ils peuvent être mis à profit pour faire travailler les programmes background au remplissage de la queue d'entrée ou au vidage des queues de sortie.

Pour pallier cet inconvénient, une version de DISK SCOPE effectuant le rollin-rollout des programmes centraux a été envisagée et partiellement réalisée [2]. Un tel projet implique des modifications dans l'ensemble du système DISK SCOPE.

Avec un tel système, une lecture sur le clavier alphanumérique pourrait s'effectuer de la manière suivante :

- un programme A fait une requête pour une lecture sur le clavier;
- le système lance l'opération par l'intermédiaire du driver;
- le système évacue le programme A sur le disque, charge à partir de celui-ci le programme B et lui donne le contrôle;

- au moment où l'utilisateur donne le signal de fin de message, le programme A est rappelé et remplace B.

Le rollin-rollout implique des modifications dans les drivers : ceux-ci n'ont plus la possibilité de travailler directement sur la zone de l'utilisateur, si c'était le cas, dans l'exemple ci-dessus, le programme B serait partiellement détruit par la lecture demandée par le programme A. Elle doit donc se faire dans une zone réservée par le driver, d'où une limitation de la longueur des messages. Le signal de fin de message provoque le rappel de A puis la transmission du contenu de la mémoire dans la zone stipulée dans la requête. Un procédé symétrique serait applicable à l'écriture.

Une autre modification est nécessaire : après le transfert d'un caractère, un retour au système, par DRSTC, s'impose avant d'initialiser le transfert suivant. Pour qu'un échange de programmes puisse avoir lieu, aucune interruption ne doit être en attente, ce passage dans INC après réception de toutes les interruptions rend un échange possible à ce moment là. Malgré les désavantages mentionnés ci-dessus, quelques programmes centraux utilisent un ou plusieurs équipements du terminal.

Il faut tout d'abord citer les programmes de tests, qui sont plus faciles à modifier ou à adapter en tant que programmes centraux plutôt que comme programmes background.

Enfin, un programme de dépouillement d'élections, écrit en FORTRAN, utilise le dispositif de visualisation. Dans cette application, il sert essentiellement à afficher des messages d'erreurs ou des indications générales, remplaçant ainsi avantageusement la machine à écrire de la console.

L'utilisation du caractère de contrôle de l'impression 1 permet une temporisation : l'opérateur a la possibilité d'effectuer une correction en cas d'erreur et le programme ne se poursuit qu'au moment où le bouton d'interruption manuelle a été enclenché.

9.4 Routines auxiliaires

Bien que le programmeur travaillant en FORTRAN ait accès à toutes les commandes du dispositif de visualisation et du clavier, quelques routines en assembleur sont nécessaires et facilitent sa tâche.

Ces routines ont une séquence d'appel spéciale en ce sens que le numéro logique de l'unité référencée n'y figure pas. Chacune de ces routines étant prévue pour un type d'équipement, elle fait appel à une routine auxiliaire en assembleur, LUN., qui recherche le numéro logique de l'unité dont elle reçoit le mnémonique.

Pour cette recherche, LUN. parcourt les tables du système comme le fait l'overlay KBDYL. L'adjonction d'une requête a été nécessaire à cet effet : SYSTAB indique au programme ayant fait la requête les adresses des tables du système les plus importantes. Ce type de recherche n'est applicable que s'il n'existe qu'une seule unité répondant au mnémonique donné, sinon LUN. indique le numéro logique de la première unité assignée, du type cherché.

Si deux claviers identiques étaient utilisés par le même programme, il serait alors nécessaire de faire figurer le numéro logique dans les séquences d'appel.

Routines à l'usage du dispositif de visualisation

INTSIZ (I,T)

Cette sous-routine fixe une nouvelle valeur de l'intensité I et de la taille T ($I \leq 1$, $T \leq 7$).

LINE (I1,J1,I2,J2)

est une sous-routine traçant le segment de droite joignant les points (I1,J1) et (I2,J2). Les coordonnées sont entières et désignent des adresses absolues sur l'écran.

CLINE (I1,J1,I2,J2)

efface la droite tracée par LINE (I1,J1,I2,J2).

Routines à l'usage du clavier

KBTAB (T,N)

A l'aide de cette sous-routine, l'utilisateur indique au driver les caractéristiques de la table de mots-clés. T est son origine, N la longueur commune des mots-clés.

KBLIGHT (L1,L2,...,Lm)

est la sous-routine effectuant l'allumage des lampes de sélection. Certes, l'utilisateur peut parvenir au même résultat sans passer par KBLIGHT, mais celle-ci lui évite le codage. Le nombre d'arguments de KBLIGHT est variable, il faut que $0 \leq L_i \leq 64$, si la valeur k figure parmi les paramètres, la lampe k est allumée, si la valeur 0 y figure, l'extinction préalable de toutes les lampes a lieu.

Tous ces sous-programmes, à l'exception de KBTAB, ne font pas directement leurs requêtes d'écriture mais s'adressent aux routines d'entrée-sortie du FORTRAN, qui à leur tour appellent IOP. (input-output processor) qui contient toutes les requêtes à DISK SCOPE. Pour des raisons de mise à jour de tables, ces routines d'entrée-sortie exigent que tous les appels au système passent préalablement par IOP.

Pourtant, KBTAB effectue sa requête directement, car IOP. n'offre pas la possibilité de travailler avec un mot de contrôle IOS_W^R , nécessaire en l'occurrence. Comme il n'y a pas de transfert d'information vers le périphérique, cette entorse ne cause pas d'ennui.

Si, lors de l'appel aux routines décrites ci-dessus, des paramètres sont hors des limites permises, des messages d'erreur sont écrits, ils ne provoquent pas la terminaison du travail, mais les valeurs incorrectes sont ignorées.

9.5 Utilisation par différents langages

Des essais pour utiliser le terminal sous le contrôle d'interpréteurs et d'un compilateur ont également été tentés. Ces applications n'ont pas posé de problème pour les entrées-sorties.

FORTRAN

Le compilateur FORTRAN accepte les instructions données par la machine à écrire ou le clavier, mais il est prévu exclusivement pour le traitement par lots. Cela veut dire qu'une instruction transmise ne peut plus être corrigée. Sous cette forme, cette application ne présente qu'un intérêt de démonstration.

Le passage à travers un programme background éditeur est un moyen de rendre cette application plus rationnelle.

LISP

Les instructions à l'interpréteur LISP peuvent être envoyées à partir du terminal.

A cause de son grand nombre de mots usuels, LISP a été une des premières applications utilisant le clavier de fonctions. Il a suffi d'ajouter à l'interpréteur la définition des mots-clés. Pour les éléments ne figurant pas sur le clavier de fonctions, il faut compléter à l'aide du clavier alphanumérique.

L'interpréteur LISP est capable de lire les données en format libre et le passage d'un clavier à l'autre ne pose aucun problème pour le traitement des données. Pour l'utilisateur, par contre, le passage du clavier alphanumérique au clavier de fonctions requiert une certaine habitude, même si les mots-clés sont rangés par ordre alphabétique.

Dans un programme LISP d'une certaine envergure, il est nécessaire de définir un certain nombre d'expressions à l'aide de la pseudo-fonction DEFINE. Ces définitions sont possibles à partir du clavier, les corrections le sont également puisqu'une fonction peut être redéfinie à tout moment. Le désavantage réside dans la longueur des définitions. Un moyen de tourner cette difficulté est prévu : il consiste à faire lire les définitions à partir du lecteur de cartes et d'introduire dans le système LISP deux fonctions : READKB et WRITEDY qui ne sont en fait que des extensions des fonctions READ et PRINT prévues pour les unités d'entrée et de sortie standard seulement. De cette manière, le système LISP pourra recevoir, à l'intérieur

d'un même programme, des données de deux sources différentes, seuls les appels aux fonctions définies sur cartes se feront sur le clavier.

BASIC

Un interpréteur BASIC restreint a été écrit au CICE pour l'ordinateur CDC 3800 [3].

Du point de vue interactif, il présente deux qualités intéressantes :

- Il peut fonctionner soit comme programme central, soit comme programme background, le passage de l'un à l'autre dépend d'un paramètre d'assemblage. Pour la mise au point, le programme central est préférable; pour l'exploitation, le programme background implique un meilleur emploi du temps mais une place en mémoire plus limitée.
- L'interpréteur BASIC possède un petit éditeur incorporé. De cette manière la correction d'instructions est possible à tout moment.

INFOL

C'est essentiellement pour le développement d'un langage interactif d'interrogation de fichiers que le terminal CICE a été conçu. Ce langage appelé INFOL (information oriented language), créé pour les ordinateurs CDC 3600 - 3800, a été écrit en assembleur COMPASS.

L'idée et une partie des définitions ont été reprises à l'Université Northwestern (Chicago) dans l'écriture d'un INFOL en FORTRAN.

S'inspirant de ces travaux, un groupe du CICE a entièrement réécrit INFOL en FORTRAN standard, en élargissant ses possibilités [4]. Deux versions ont été développées :

- une version standard, non interactive, portable, dans laquelle l'utilisateur a la possibilité d'introduire ses propres routines en FORTRAN [5];
- une version interactive, également portable [6].

La compatibilité des deux versions est très poussée.

La seconde version se distingue de la première par un comportement différent en cas d'erreur : d'autres messages apparaissent et la récupération est toujours possible.

L'utilisateur peut travailler en INFOL aussi bien sur la machine à écrire que sur l'ensemble clavier-écran, il doit cependant préciser les dimensions de la page. INFOL utilise toutes les possibilités du clavier de fonctions, y compris les lampes de sélection, c'est là le seul point spécifiquement adapté aux possibilités de l'installation de l'Université de Genève. Toutes les lectures se font en format libre, les écritures tiennent compte du format de page indiqué. En plus de ce qui apparaît sur le terminal, INFOL permet d'obtenir des copies sur l'imprimante.

Les possibilités d'INFOL et celles de l'ensemble clavier-écran se complètent de manière intéressante. INFOL constitue un programme efficace et fiable. L'inconvénient est qu'il travaille en tant que programme central, même si des programmes background fonctionnent simultanément, les pertes de temps sont non négligeables. La mise au point de rollin-rollout permettra une utilisation plus rationnelle du temps machine.

Chapitre 10 : Extensions

10.1 Modifications des équipements

L'écriture des drivers pour l'installation de l'Université de Genève a été facilitée par le fait qu'il y a une seule machine à écrire, un seul écran et un seul clavier, chacun constituant à la fois un équipement et une unité.

Comment serait-il possible d'en augmenter le nombre ?

Quelles seraient les implications pour le hardware, le système et les drivers ? Elles font l'objet des trois premières sections de ce chapitre. Il est évident qu'il y a plusieurs solutions possibles et que toute modification dans l'un des domaines peut en entraîner dans les autres.

Tout d'abord sur le plan du hardware, deux possibilités d'extension apparaissent :

- ou bien on maintient la formule actuelle d'une unité par équipement, la configuration actuelle pourrait être augmentée d'environ 4 équipements (c'est-à-dire 4 unités);
- ou bien alors on envisage des équipements comportant plusieurs unités, ce qui offre une augmentation plus importante.

La première question qui se pose au sujet de cette deuxième solution est son opportunité : elle entraînerait une charge supplémentaire non négligeable pour l'unité de calcul et les canaux et il serait certainement bien meilleur de relier toutes les nouvelles unités à un mini-ordinateur et de connecter celui-ci à un canal.

D'autre part si, pour le hardware, on désire conserver la conception actuelle des équipements, la seule solution est la première : construire des équipements analogues à ceux déjà existants.

Donc, au cas où on désirerait augmenter le nombre des unités interactives, il serait judicieux de se limiter à la première solution. Toutefois, il sera intéressant d'examiner les modifications du software liées à chacune des deux possibilités d'extension.

A la conception de la partie électronique de la machine à écrire, il n'y aurait pas de modification importante à apporter. Par contre, une modification fondamentale devrait être envisagée pour le clavier et le dispositif de visualisation : ils ne devraient constituer au point de vue physique et également au point de vue logique qu'un seul et même équipement.

Ceci impliquerait quelques changements dans les codes de fonctions et de statut; les opérations de lecture ne seraient pas affectées. Pour les opérations d'écriture, il serait alors nécessaire, avant chaque instruction BEGW, d'envoyer une fonction indiquant si les données sont destinées à l'écran ou au clavier (allumage des lampes).

La conséquence pour le software serait qu'il n'y aurait plus 2 drivers, un overlay et la partie résidente associée, mais un seul driver traitant un plus grand nombre d'opérations. L'affichage simultané se ramènerait à une simple question de communication entre les différentes parties du driver.

Ainsi seraient supprimés les chargements de KBDYL à partir du disque ainsi que les recherches effectuées par cet overlay dans les tables afin de retrouver l'unité DY. De même, l'unité DYS au niveau 63 n'aurait plus de raison d'être.

La construction d'un tel ensemble représente un travail considérable. Dans une telle situation, le système de simulation est capable de rendre des services appréciables : le driver pourrait être développé parallèlement à la construction de la partie électronique; le règlement des questions qui se posent inévitablement dans la coordination des tâches hardware et software s'en trouverait facilité.

10.2 Implications pour le système d'exploitation

Si, sur le plan du software, la majorité des modifications concerne les drivers, quelques points relatifs au reste du système sont également à considérer.

Tout d'abord les tables. La solution adoptée pour le cas particulier du terminal du CICE est optimale ou presque en ce qui

concerne la place en mémoire. Elle utilise plus de positions auxiliaires que nécessaire, mais leur traitement dans les drivers est plus aisé et prend moins de place.

Maintenir la même solution reviendrait à multiplier la longueur actuelle des tables auxiliaires par le nombre d'unités : le gain de place dans le driver ne compenserait plus, à partir d'un certain nombre d'unités, celle perdue dans les tables. Pour le cas général, il est plus avantageux d'utiliser au mieux chaque mot des tables et d'y concentrer l'information, quitte à allonger les drivers.

Au début de 1973, un nouveau driver des bandes magnétiques a été introduit. Il comporte deux niveaux : le premier effectue les opérations, tandis que le niveau supérieur les contrôle et applique les algorithmes de récupération en cas d'erreur. Pour ce driver, le problème des tables auxiliaires s'est également posé et a été résolu de manière intéressante : 4 tables parallèles à AET ont été introduites : EER1, EER2, EER3, EER4. Ces tables sont générées par PREDRUM et leur nombre peut être augmenté par un paramètre. Chaque unité physique dispose d'une entrée dans EER1, EER2, EER3, EER4. Pour l'adressage, l'ordinal AET (IORD) est également l'ordinal EERn; quant à l'utilisation, elle est absolument libre. Il convient de remarquer que les cellules XXAUX occupent des positions de mémoire consécutives et que leur nombre est adapté au type d'équipement, tandis que les cellules EERn d'une même unité sont réparties dans plusieurs tables et que leur nombre ne dépend que d'un paramètre d'installation. Ainsi, mêmes les unités qui n'en ont pas besoin ont des places réservées dans les tables EERn. Pour le cas général, les tables EERn sont cependant plus pratiques que les tables XXAUX et on peut donc envisager de remplacer celles-ci par celles-là. Le détail de ce remplacement ainsi que celui des tables XXCW sont examinés dans la section 10.3.

Un deuxième point est l'allongement des drivers, d'une part pour les raisons énoncées ci-dessus et d'autre part pour la concentration de DRDYD et de DRKBD en un seul driver.

L'allongement des drivers implique celui de BOVER qui occupe de la place même lorsque le terminal n'est pas utilisé. Cette zone sert entre autres à un chargement momentané du driver afin de traiter les instructions que l'opérateur peut donner à la console. Or, toutes celles de ces instructions qui ont un sens pour les équipements du terminal se ramènent à un appel au driver avec le paramètre STA. Il est donc tout à fait possible d'envisager, à côté du driver complet, un driver restreint destiné à traiter les instructions de l'opérateur et ne reconnaissant que la fonction STA. La longueur de BOVER pourrait alors être calculée d'après celle des drivers restreints et serait ainsi sensiblement diminuée.

Si l'on s'en tient, pour le terminal, au principe "une unité par équipement" il n'y a pas d'autre modification à envisager dans le système. Si par contre, plusieurs unités sont rattachées à un même équipement, il faut tenir compte d'une restriction énoncée dans la section 3.6. XCONN refuse de connecter une unité si une autre unité attend déjà une interruption à travers le même canal et le même équipement (même si le canal n'est pas actif en ce moment-là). Cela revient à dire que deux claviers, rattachés au même équipement, ne pourraient pas attendre simultanément une interruption sur caractère prêt alors que cela est physiquement possible.

La raison de cette restriction est la suivante : au moment d'une interruption sur un canal, l'instruction CPJ permet de déterminer l'équipement d'où elle provient. Par contre, il n'existe pas d'instruction analogue au niveau de l'équipement et la seule solution consiste à tester successivement les unités susceptibles d'avoir envoyé l'interruption. En ne permettant à un instant donné qu'une seule unité attendant une interruption sur un même équipement, le système évite toute ambiguïté dans ICHAN. On pourrait donc supprimer la restriction dans XCONN mais alors, il faut que le driver soit capable de distinguer si une unité susceptible d'avoir envoyé une interruption l'a réellement envoyée ou s'il faut poursuivre la recherche. Pour que cette distinction soit possible, à

l'aide de l'instruction COPY, il suffit qu'un bit de statut soit associé à chaque condition d'interruption (ce qui est le cas dans la version actuelle).

Le cas où deux interruptions arriveraient simultanément ne pose pas non plus de problème : l'unité ayant le plus grand ordinal AET serait traitée la première; pendant ce traitement l'interruption relative à l'autre unité ne risquerait pas d'être effacée puisqu'aucune instruction CLCH n'est exécutée.

10.3 Modifications des drivers

Le principal problème pour les drivers est de définir une nouvelle répartition des chaînes de mots de contrôle et des variables auxiliaires dans les tables EERn.

Les zones de mots de contrôle jouent un rôle double : si au cours de l'exécution d'une chaîne l'un des mots de contrôle est volontairement ou non modifié, le système reste totalement protégé et, pour le programme de l'utilisateur, tout se passe comme si la chaîne n'avait pas été modifiée.

De ces deux points, le premier est essentiel, le second est accessoire : si l'on renonce à celui-ci, la suppression des zones XXCW peut se faire de la manière suivante : l'appel à MOV CW est remplacé par un appel à FET CW, le mot de contrôle en cours est stocké dans USTA et son adresse dans USTQ.CWA. Le bit USTQ.B reprend le rôle de flag joué par l'adresse inférieure du premier mot de XXCW, il est mis à 0 à la fin de chaque opération et à 1 au début de chaque opération. A la fin du traitement de chaque mot de contrôle a lieu un nouvel appel à FET CW. De cette manière, la protection totale du système est maintenue.

Ceci constitue une dérogation aux principes généraux de DISK SCOPE, légère puisque l'intégrité du système reste assurée, d'ailleurs, le driver de la machine à écrire de la console procède de cette même manière.

Pour la suppression des autres variables, il convient de distinguer les utilisations que le driver en fait. Il y a trois

sortes de variables auxiliaires :

- 1) Celles situées à l'intérieur du driver définies en cours de traitement et reprises avant la sortie. Par exemple, dans DRDYD, le byte en cours de traitement par CTRL est de ce type.
- 2) Celles situées à l'intérieur du driver, définies en cours de traitement et reprises lors d'une entrée ultérieure, mais avant la fin de l'opération. L'utilisation de ces variables n'est possible que si la condition empêchant le déplacement du driver ($AET.L \neq 0$) est vérifiée aussi longtemps que la variable doit être conservée. ENDPAGE et ENDBUF, dans DRDYD, sont de ce type.
- 3) Les valeurs qui gardent une valeur pendant plusieurs opérations. Ce sont celles qui sont placées dans les tables et dans les zones XXAUX.

Dans les généralisations envisagées, les variables de type 1) peuvent être maintenues et même augmentées en nombre. Celles de type 2) doivent être remplacées par des variables de type 3) : en effet, le driver peut lancer successivement des opérations sur deux équipements différents et il ne doit se produire aucune interférence de l'un sur l'autre. Quant aux variables de type 3), elles doivent être replacées dans les tables EERn. Il suffit donc de reprendre toutes les variables des types 2) et 3), définies pour chacun des drivers, et voir comment les concentrer dans les mots disponibles.

Machine à écrire

Les variables sont (voir section 6.2) .

nom	signification	longueur maximale (en bits)
BLCOUNT	compteur d'espaces	18
CASE	option de l'utilisateur pour la position du clavier en écriture	1
FSTCHAR	indicateur du premier caractère significatif en lecture	1
INDX	2 pointeurs dans la zone de l'utilisateur : un pour le mot, l'autre	

pour le caractère à l'intérieur de
celui-ci 15 + 6

Comme la zone TWAUX comporte 2 mots et que les tables EERn en offrent 4, il n'y a pas de problème.

Clavier et dispositif de visualisation

Le compte ci-dessous est fait pour un driver contrôlant des unités du type clavier-écran. Pour la partie écran, les variables sont (voir section 7.2) :

nom	signification	longueur maximale (en bits)
HADDR	valeurs des registres du dispositif de visualisation	12
VADDR		12
HMARGIN		8
VMARGIN		8
CONTROL		8 + 8
SIZE	information redondante	0
BLCOUNT	compteur d'espaces	18
INDX	comme pour DRTWD	15 + 6
ENDBUF	flag de fin de la zone DYBUF	1
ENDPAGE	flag de fin de page	1
KBDY	interaction clavier-écran	1

Pour la partie clavier, les variables sont (voir section 8.2) :

nom	signification	longueur maximale (en bits)
TABLE	informations relatives à la table des mots-clés	18 + 6
INDX	comme pour DRTWD	15 + 6
CODE	informations pour KBDYL deviennent des variables de type 1) en vertu de la nouvelle conception du driver	0

La longueur du mot-clé est limitée ici à 63 mots machine.

Le total donne 143 bits alors que 192 sont disponibles, il y a donc de la place en suffisance. Il est avantageux, à chaque entrée dans le driver, de prendre l'information stockée dans ces

4 mots et de la répartir dans un plus grand nombre de variables (du type 1), ce qui facilite leur accès pour les différentes opérations, et à chaque sortie du driver de concentrer les nouvelles valeurs de ces différentes variables dans les tables EERn.

Dans les versions actuelles de DRDYG et de DRKBD, les requêtes WRITE ont des significations bien précises. Dans le cas d'un équipement du type clavier-écran, ces significations ne peuvent être maintenues car elles introduiraient une ambiguïté : comment le driver peut-il distinguer une écriture sur l'écran d'un allumage de lampes ? Cette ambiguïté peut être levée de plusieurs manières, en voici une :

Requête	Fonction associée	Effet
WRITE	WRF	écriture sur l'écran
LOCATE (avec mot de contrôle IOS _W ^R)	LOC	définition d'une table de mots-clés
LOCATE (avec mot de contrôle IOT _W ^R)	LOC	allumage des lampes

Une autre modification concernant les deux drivers a déjà été abordée dans la section 9.3. Dans la version actuelle, après la lecture ou l'écriture d'un caractère sur le clavier ou la machine à écrire, le driver commande immédiatement le début du traitement du caractère suivant. Ce procédé se justifie parfaitement dans le cas d'un seul équipement et d'une seule unité, mais dès que le driver commande plusieurs équipements ou unités, une modification s'impose : après le traitement d'un caractère, le driver doit retourner à DRSTC afin de desservir une unité qui serait en attente.

Certes, l'ensemble de ces modifications n'est pas négligeable, mais il faut relever qu'elles affectent peu le déroulement logique des opérations; celui-ci peut être conservé presque intégralement.

10.4 Conclusion

Le problème de l'écriture du software pour de nouveaux équipements est, par certains côtés rigide, et par d'autres laisse passablement de liberté.

Rigide, parce que la réalisation est forcément limitée à ce dont le hardware est capable. Rigide, également, car les drivers doivent respecter les contraintes fixées par la philosophie du système d'exploitation.

La liberté réside dans le choix des possibilités d'action et des commodités offertes à l'utilisateur. Elle est toutefois restreinte par le respect de la compatibilité avec les autres équipements. La programmation elle-même laisse passablement de liberté grâce au jeu très riche d'instructions de l'ordinateur CDC 3800. "Comment interpréter tel mot de contrôle dans telle situation ?", "Quelle est la façon la plus rationnelle de construire une table de conversion de codes ?", voilà deux exemples de questions ayant plusieurs réponses. Il convient évidemment de choisir celle qui est la plus économique en temps et en place.

Pour le software, la qualité de résultat final dépend du système d'exploitation. Il est évident qu'avec DISK SCOPE, il ne vient pas en question d'atteindre les performances d'un ordinateur de la troisième génération, tant les conceptions de base sont différentes.

Implanter un terminal sur un ordinateur n'ayant pas le temps partagé, c'est admettre par avance une certaine perte de temps due au délai de réponse de l'utilisateur devant son clavier. Ce terminal représentait cependant une nécessité pour un Institut universitaire développant un langage interactif, tel INFOL. Ne serait-ce que pour cette raison, ce travail en aura valu la peine.

Annexe 1 : Quelques instructions COMPASS

A côté des instructions d'entrée-sortie décrites dans la section 2.2, quelques instructions de base sont utilisées dans ce travail.

Notations

(a)	terme de bank
b	registre d'indice Bb, b = 7 désigne l'adressage indirect
i	terme de bank
m	adresse en mémoire
y	opérande
(A), (Q), (Bb)	contenu du registre A, Q, Bb.

Instructions

ENA	y,b	(enter A)
ENQ	y,b	(enter Q)

la valeur y, modifiée par Bb, est placée dans le registre A ou Q.

LDA	m,b	(load A)
LDQ	m,b	(load Q)

le contenu de l'adresse m, modifiée par Bb, est chargé dans le registre A ou Q.

STA	m,b	(store A)
STQ	m,b	(store Q)

le contenu de A ou de Q est stocké à l'adresse M ($M = m + (Bb)$).

DLDA	m,b	(load A double precision)
------	-----	---------------------------

le contenu des adresses M et M + 1 est chargé dans les registres A et Q.

DSTA	m,b	(store A double précision)
------	-----	----------------------------

est l'opération inverse.

SLJ	m	(selective jump)
-----	---	------------------

saut inconditionnel à l'adresse m.

RTJ m (return jump)

saut avec retour à l'adresse m.

Les instructions SLJ et RTJ se font nécessairement à l'intérieur d'un même bank.

UBJP (a)m,b,i (unconditional bank jump)

saut inconditionnel à l'adresse (a)m + (Bb), la valeur i est placée dans OB.

BRTJ (a)m,b,i (bank return jump)

saut avec retour à l'adresse $M = (a)m + (Bb)$, OB prend la valeur de i. En fait, le hardware stocke une instruction de retour UBJP à l'adresse M et donne le contrôle à l'adresse $M + 1$. UBJP et BRTJ sont des instructions permettant de modifier IB.

Si $a = m = b = i = 0$, le système d'exploitation DISK SCOPE interprète cette instruction comme une requête. Le code d'appel est contenu dans une portion de mot non utilisée par l'instruction.

EXEC (a)m,b (execute)

exécute l'instruction à l'adresse $M = (a)m + (Bb)$, sans modifier la valeur du compteur d'instructions.

La pseudo-instruction

s EQU v

donne la valeur v au symbole s.

Annexe 2 : Les codes des caractères

Remarque : dans toute l'annexe, les codes sont en octal.

Machine à écrire : code IBM Selectric

code	maj.	min.	code	maj.	min.	code	maj.	min.
00			32	.	.	55	D	d
01	B	b	33	#	3	56	R	r
02	W	w	34	+	=	57	%	5
03			35	U	u	60	espace	
04	—	-	36	V	v	61	BS	BS
05			37	@	2	62	tabulateur	
06			40	Y	y	63	CR	CR
07	(	9	41	L	l	64	majuscule	
10	Q	q	42	O	o	65	ruban rouge	
11	C	c	43	)	0	66	minuscule	
12	A	a	44	?	/	67	ruban noir	
13	¢	6	45	H	h	70	J	j
14	,	,	46	S	s	71	X	x
15	K	k	47	\$	4	72	M	m
16	I	i	50	:	;	73	Z	z
17	*	8	51	E	e	74	G	g
20-27			52	"	'	75	T	t
30	F	f	53	&	7	76	$\frac{1}{4}$	$\frac{1}{2}$
31	N	n	54	P	p	77	±	1

Abréviations :

BS : recul

CR : retour de chariot

Dispositif de visualisation : code ASCII à 6 bits

code	car.	code	car.	code	car.	code	car.
00	■	20	P	40	@	60	0
01	A	21	Q	41	!	61	1
02	B	22	R	42	"	62	2
03	C	23	S	43	#	63	3
04	D	24	T	44	\$	64	4
05	E	25	U	45	%	65	5
06	F	26	V	46	&	66	6
07	G	27	W	47	'	67	7
10	H	30	X	50		70	8
11	I	31	Y	51	.	71	9
12	J	32	Z	52	*	72	:
13	K	33	[	53	+	73	;
14	L	34	\	54	,	74	<
15	M	35	]	55	-	75	=
16	N	36	{	56	.	76	>
17	O	37	}	57		77	?

Fonctions :

160 : espace

161 : recul

163 : retour au début de la ligne suivante

164 : remise à zéro du registre d'adresse verticale

165 : espacement vertical

Clavier alphanumérique : code ASCII à 7 bits

code	car.	code	car.	code	car.	code	car.
000	NUL	054	,	110	H	144	d
001	SOH	055	-	111	I	145	e
002	STX	056	.	112	J	146	f
003	ETX	057	/	113	K	147	g
004	EOT	060	0	114	L	150	h
005	ENQ	061	1	115	M	151	i
006	ACK	062	2	116	N	152	j
007	BEL	063	3	117	O	153	k
010	BS	064	4	120	P	154	l
011	HT	065	5	121	Q	155	m
012	LF	066	6	122	R	156	n
013	VT	067	7	123	S	157	o
014	FF	070	8	124	T	160	p
015	CR	071	9	125	U	161	q
016	SO	072	:	126	V	162	r
040	espace	073	;	127	W	163	s
041	!	074	<	130	X	164	t
042	"	075	=	131	Y	165	u
043	#	076	>	132	Z	166	v
044	\$	077	?	133	[	167	w
045	%	100	@	134	\	170	x
046	&	101	A	135	]	171	y
047	'	102	B	136	^	172	z
050	(	103	C	137	_	173	{
051	)	104	D	140	`	174	
052	*	105	E	141	a	175	}
053	+	106	F	142	b	176	~
		107	G	143	c		

Pour les caractères spéciaux, les abréviations liées au code ASCII n'ont pas leur sens usuel. Seuls, certains sont utilisés pour des fonctions sur l'écran (voir section 8.3).

Correspondance des caractères sur les différents équipements
en mode BCD

BCD	LP	TW	DY	KB	BCD	LP	TW	DY	KB
00	0	0	0	0	40	-	-	-	-
01	1	1	1	1	41	J	J	J	J
02	2	2	2	2	42	K	K	K	K
03	3	3	3	3	43	L	L	L	L
04	4	4	4	4	44	M	M	M	M
05	5	5	5	5	45	N	N	N	N
06	6	6	6	6	46	O	O	O	O
07	7	7	7	7	47	P	P	P	P
10	8	8	8	8	50	Q	Q	Q	Q
11	9	9	9	9	51	R	R	R	R
12	:	:	:	:	52	V			
13	=	=	=	=	53	\$	\$	\$	\$
14	≠	'	'	'	54	*	*	*	*
15	≤	?	?	?	55	↑	UC	{	{
16	%	%	%	%	56	↓	LC	}	}
17	[		[	[	57	>		>	>
20	+	+	+	+	60	espace		espace	
21	A	A	A	A	61	/	/	/	/
22	B	B	B	B	62	S	S	S	S
23	C	C	C	C	63	T	T	T	T
24	D	D	D	D	64	U	U	U	U
25	E	E	E	E	65	V	V	V	
26	F	F	F	F	66	W	W	W	W
27	G	G	G	G	67	X	X	X	X
30	H	H	H	H	70	Y	Y	Y	Y
31	I	I	I	I	71	Z	Z	Z	Z
32	<	BS	<	<	72	]	"	]	]
33	.	.	.	.	73	,	,	,	,
34	)	)	)	)	74	(	(	(	(
35	≥	±	!	!	75	→	TAB	@	@
36	↵			—	76	≡	—	≡	~
37	;	;	;	;	77	∧	#	#	#

Abréviations :

BCD : code BCD interne

LP : caractère sur l'imprimante

TW : caractère sur la machine à écrire

DY : caractère sur l'écran

KB : caractère sur le clavier alphanumérique

BS : recul (backspace)

UC : passage en majuscule (upper case)

LC : passage en minuscule (lower case)

TAB : tabulateur

Les cases vides correspondent à des caractères qui sont ignorés dans la routine de conversion.

Annexe 3 : Termes anglais

Plutôt que des définitions rigoureuses, cette annexe contient des explications de quelques termes anglais utilisés dans le texte.

Hardware (quincaillerie) :

Tout le matériel constituant l'ordinateur, en particulier sa partie électronique.

Software :

Ensemble des programmes permettant l'utilisation de l'ordinateur.

Pour remplacer ces deux termes, les mots "matériel" et "logiciel" ont été adoptés en français.

Bit :

Unité élémentaire d'information, il ne peut prendre que les valeurs 0 et 1.

Byte :

Dans la terminologie IBM : 8 bits, soit un caractère.

Dans la terminologie CDC : un nombre variable de bits, n'excédant pas un mot.

Flag :

Variable pour laquelle deux états sont pris en considération (usuellement : 0 et $\neq 0$). Le terme français de sémaphore a été adopté.

Overlay :

Portion de programme pouvant être chargée temporairement dans une zone fixe de la mémoire. Les overlays du système DISK SCOPE sont des programmes dont l'utilisation peu fréquente ne justifie pas l'implantation permanente en mémoire.

Dump :

Impression de toute une zone de mémoire. Le système DISK SCOPE fait des dumps sur demande en cours ou à la fin d'un travail.

Les termes suivants sont expliqués dans le texte, lors de leur première apparition, entre parenthèses est indiqué le numéro de la section.

Autoload (1.1);
Bounds register (2.1);
Bank (2.1);
Operand bank register (2.1);
Instruction bank register (2.1);
Background (3.2);
Bypass (3.3);
Driver (3.5).

Pour la notion de programme background, le terme français de symbiont a été proposé; son usage reste cependant peu courant.

Annexe 4 : Références

Pour la rédaction du chapitre 1, les sources suivantes ont été utilisées :

Manson, N.

Traité pratique d'informatique.

- Paris, Techniques de l'Ingénieur, 1970;

Steel, T.B.

Multiprogramming - Promises, performance and prospect.

Dans : AFIPS conference proceedings.

33 (1968) part 1, pp. 99-103.

- Washington, Thompson Book Company, 1968;

Denning, P.J.

Third generation computer systems.

Dans : ACM computing surveys.

3 (1971) 4, pp. 175-216.

- New York, Lee Revens, 1971;

Les équipements du terminal sont décrits dans des notices internes, concernant respectivement le simulateur de canal, la machine à écrire, le dispositif de visualisation et les claviers.

Pour la programmation, les ouvrages suivants ont servi de base :

Gineste, M.; Guiho, G.; Prost, A.; Vignes, J.

Ordinateur CDC 3600, le langage COMPASS et le système.

- Paris, Technip, 1972;

3800 computer system reference manual.

- St. Paul, CDC, 1968; (No 60163200);

3000 series peripheral equipment reference manual.

- St. Paul, CDC, 1968; (No 60108800);

3600 computer systems COMPASS reference manual.

- Palo Alto, CDC, 1966; (No 60052500C);

3400/3600/3800 COMPASS training manual.

- Minneapolis, CDC, 1967; (No 60166700);

3600/3800 computer systems DRUM SCOPE reference manual.

- St. Paul, CDC, 1968; (No 60059200B);

3600/3800 DRUM SCOPE internal maintenance specifications.

- S.L., CDC, 1967.

Il convient de mentionner également l'ensemble des programmes constituant le système d'exploitation DISK SCOPE.

Le chapitre 7 se réfère à l'article suivant :

[1] Vuilleumier, G.

La place des systèmes d'affichage à balayage séquentiel dans les communications homme-machine.

Dans : Compte rendus des journées d'électronique 1972, p. 213-224.

- Lausanne, EPFL, 1972.

Le chapitre 9 cite les documents suivants :

[2] Harms, J.

Rollin-rollout.

- S.L.N.D.; [communication personnelle];

[3] Pellegrini, C.

Programme BASIC.

- Genève, IICE, 1972;

[4] INFOL-2 manuel de référence. 2me édition.

- Genève, Centre interfacultaire de calcul électronique, 1974.

[5] Chenais, J-M.

Etude de la portabilité d'INFOL-2.

Thèse de doctorat, n° 1616.

- Genève, CICE, 1973.

[6] Baud, J-P.

Un système interactif pour la création et l'interrogation de fichiers. Thèse de doctorat, n° 1620.

- Genève, CICE, 1974.

